Uncertainty Estimation, Management, and Utilisation in Human-Computer Dialogue

Inaugural-Dissertation

zur Erlangung des Doktorgrades
der Mathematisch-Naturwissenschaftlichen Fakultät
der Heinrich-Heine-Universität Düsseldorf

vorgelegt von

Carel van Niekerk
aus Südafrika

Düsseldorf, Oktober 2023

aus dem Institut für Informatik
der Heinrich-Heine-Universität Düsseldorf

Gedruckt mit der Genehmigung der
Mathematisch-Naturwissenschaftlichen Fakultät der
Heinrich-Heine-Universität Düsseldorf

Berichterstatter:

1. Prof. Dr. Milica Gašić

2. Prof. Dr. Josef van Genabith

Tag der mündlichen Prüfung: 18. März 2024

Declaration of Authorship

I, *Carel van Niekerk*, declare under oath that I have produced my thesis independently and without any undue assistance by third parties under consideration of the Principles for the Safeguarding of Good Scientific Practice at Heinrich Heine University Düsseldorf.

Düsseldorf,

Location, Date	Carel van Niekerk

Acknowledgements

I would like to start by thanking God, for gifting me with the health, strength, and talents necessary to undertake and complete this journey.

I would like to extend special thanks to my supervisor, Prof. Dr. Milica Gašić, whose invaluable support and motivation were pivotal throughout this journey. Her patience, support, positivity, and insightful feedback have been instrumental in my ability to navigate and surmount the challenges encountered along the way.

I would also like to thank the Alexander von Humboldt Foundation for their generous financial support through the Sofja Kovalevskaja Award, funded by the Federal Ministry of Education and Research.

I also would like to express my thanks to my colleagues and friends in the Dialog Systems and Machine Learning Group. Their support, collaboration, and the fun moments (and pranks) we shared have been essential in maintaining my well-being and perspective during this journey.

I cannot overstate my gratitude towards my fiancée, Claudia, for her unwavering love, support, and encouragement. Her contributions, including language editing of this work, have been invaluable.

Finally, I extend my deepest thanks to my parents, Carel and Celeste, my brother Allan, and the rest of my family. Their endless support and encouragement have been my foundation throughout my academic journey.

This journey has been a collective effort, and it is with immense gratitude that I acknowledge the role each one of you has played in bringing this work to fruition.

Abstract

In the rapidly evolving field of human-computer interaction, there is an increasing demand for effective and reliable dialogue systems, computer programs engineered to converse with humans. However, these systems often fall short in unpredictable or ambiguous scenarios, a problem attributed to the absence of a comprehensive model for handling uncertainty. This limitation impacts the ability to communicate effectively with human users, thereby diminishing user experience and trust. Uncertainty is a fundamental aspect of human cognition and everyday decision-making processes, serving as both an obstacle and an opportunity in our constant pursuit of knowledge and effective communication. Despite its important role, uncertainty is underrepresented in the development of machine learning models for dialogue systems.

To address these gaps, this thesis focuses on integrating and leveraging uncertainty within task-oriented dialogue systems, systems designed to assist users in accomplishing specific tasks. With the aim of achieving human-level interactive capabilities, we make three substantial contributions to this area. First, we enhance the system's language understanding component, improving its accuracy in evaluating the certainty of its predictions. Secondly, we introduce SetSUMBT (Set Similarity based Slot Utterance Matching Belief Tracker), a model designed to capture various facets of uncertainty, bolstering the robustness and adaptability of the dialogue policy models responsible for generating system responses, as validated through simulated and real-user interactions. Thirdly, we present CAMELL (Confidence-based Acquisition Model for Efficient self-supervised active Learning with Label validation), an innovative framework which minimises the reliance of models on labelled data. By incorporating elements of self-supervision, where models learn from their own predictions, and label validation, CAMELL automates the rectification of unreliable human annotations, a feature with extensive applicability in various machine learning domains.

Incorporating insights from psychological theories on human uncertainty management, this thesis emphasises the importance of integrating such insights into machine learning models for dialogue. Our methods will advance the field by introducing more reliable, robust, and effective dialogue systems that better handle uncertainties, ultimately enhancing the quality of human-computer interaction. Furthermore, this work challenges current limitations associated with data deficiencies, offering a data-driven approach for improving dataset quality, thereby paving the way for future research in machine learning and human-computer interaction.

Contents

List of Figures

List of Tables

Summary of Notation

This chapter summarises the notations used throughout this thesis for quick reference.

General Mathematical Notations

a, b, c	Scalars
$\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}$	Vectors
a_i	i-th element of vector $\boldsymbol{a}$.
$\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}$	Matrices
$\boldsymbol{a}_j$	j-th row of matrix $\boldsymbol{A}$.
$\boldsymbol{a} = [a_i]_{i=1}^{I}$	Vector $\boldsymbol{a}$ composed of elements a_i.
$\boldsymbol{A} = [\boldsymbol{a}_i]_{j=1}^{J}$	Matrix $\boldsymbol{A}$ composed of rows $\boldsymbol{a}_i$.
$a_{i,j}$	Element in the i-th row and j-th column of matrix $\boldsymbol{A}$
$\boldsymbol{A}^\intercal$	Transpose of matrix $\boldsymbol{A}$
$\boldsymbol{A}^{-1}$	Inverse of matrix $\boldsymbol{A}$
$\boldsymbol{AB}$	Matrix multiplication of $\boldsymbol{A}$ and $\boldsymbol{B}$
$\boldsymbol{x}_{1:t} = \langle \boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_t \rangle$	Sequence of t vectors.
$\mathcal{X}, \mathcal{Y}, \mathcal{Z}$	Sets of elements.
$x \in \mathcal{X}$	x is an element of set $\mathcal{X}$.
$\boldsymbol{a} \odot \boldsymbol{b}$	Element-wise multiplication of vectors $\boldsymbol{a}$ and $\boldsymbol{b}$.
$\boldsymbol{a} \cdot \boldsymbol{b}$	Dot product of vectors $\boldsymbol{a}$ and $\boldsymbol{b}$.
$\boldsymbol{a} \oplus \boldsymbol{b}$	Concatenation of vectors $\boldsymbol{a}$ and $\boldsymbol{b}$
$\boldsymbol{a}^2 = \boldsymbol{a} \odot \boldsymbol{a}$	Element-wise square of vector $\boldsymbol{a}$.
$f(x), g(x)$	Functions dependent on variable x
$f(\boldsymbol{x}; \boldsymbol{\theta})$	Function of $\boldsymbol{x}$ parameterised by $\boldsymbol{\theta}$
$f \circ g(x) = f(g(x))$	Composition of functions f and g
$f : \mathcal{X} \to \mathcal{Y}$	Function mapping elements from set $\mathcal{X}$ to set $\mathcal{Y}$.
$\int f(x)dx$	Integral of $f(x)$
$\mathbb{R}$	The set of real numbers
$\mathbb{N}$	The set of natural numbers
$\mathbb{R}^d$	$d-$dimensional Euclidean space
$\mathbb{R}^{d_1 \times d_2}$	Real-valued matrices with d_1 rows and d_2 columns
Σ	Summation symbol
$\log(x)$	Natural logarithm of x
$e^x = \exp(x)$	Exponential function of x
$\arg\max_y f(y)$	Value of y that maximises the function $f(y)$
$\max_y f(y)$	Maximum value of function $f(y)$ with respect to y
$\mathcal{O}(n)$	Big-O notation, indicating the computational complexity of a function

Notations in Deep Learning

$\boldsymbol{x}_i$	Input feature vector for the i-th observation
$\boldsymbol{y}_i$	Target variable for the i-th observation

$(\boldsymbol{x}_i, \boldsymbol{y}_i)$	i-th observation consisting of input features and target
$\mathcal{D} = \{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=1}^{N}$	Dataset of N observations
$\phi(\boldsymbol{z})$	Activation function applied to input $\boldsymbol{z}$
$\phi^{(i)}(\boldsymbol{x})$	Activation function at the i-th layer of a deep neural network (DNN).
$g(\boldsymbol{x}; \boldsymbol{\theta})$	Neural network layer/module parameterised by $\boldsymbol{\theta}$
$g^{(i)}(\boldsymbol{x}; \boldsymbol{\theta})$	i-th layer of a deep neural network, parameterised by $\boldsymbol{\theta}$
$\boldsymbol{h}^{(i)}$	Output at the i-th layer of a deep neural network
$\mathcal{L}(\mathcal{D}; \boldsymbol{\theta})$	Loss/objective function for a model with parameters $\boldsymbol{\theta}$
$\nabla_{\boldsymbol{\theta}} \mathcal{L}(\mathcal{D}; \boldsymbol{\theta})$	Gradient of the loss function with respect to $\boldsymbol{\theta}$
η	Learning rate
$\psi(\boldsymbol{x}, \boldsymbol{y})$	Similarity or compatibility measure between $\boldsymbol{x}$ and $\boldsymbol{y}$
$\Phi(\boldsymbol{x})$	Normalization function applied to $\boldsymbol{x}$

Notations in Probability and Statistics

X	Random variable
x	A realisation of the random variable X.
$\vec{\mathsf{X}}$	Vector of random variables
$\boldsymbol{x}$	Realisation of the random vector $\vec{\mathsf{X}}$.
$\mathbf{X}$	Matrix whose elements are random variables
$\boldsymbol{X}$	Realisation of the random matrix $\mathbf{X}$.
$\mathsf{p}(\mathsf{Y} = y)$	Probability density function of the discrete random variable Y
$\mathsf{p}(\mathsf{Y}\|\cdot)$	Conditional probability of Y
$\mathsf{P}(\mathsf{Y})$	Probability density function of the continuous random variable Y
$\mathbb{E}_{\mathsf{P}\left(\vec{\mathsf{X}}\right)}\left[\vec{\mathsf{X}}\right]$	Expected value of $\vec{\mathsf{X}}$ with respect to the distribution $\mathsf{P}\left(\vec{\mathsf{X}}\right)$
$\mathsf{var}\left(\vec{\mathsf{X}}\right)$	Variance of the random variable $\vec{\mathsf{X}}$
$\mathtt{Bias}(\mathsf{X}) = \mathbb{E}[\mathsf{X}] - \theta$	Bias of an estimator X with respect to the true parameter θ
$\mathcal{H}(\mathsf{P}(\mathsf{X}))$	Entropy of the random variable X
$\mathcal{I}(\mathsf{X}, \mathsf{Y})$	Mutual Information between random variables X and Y
$D_{\mathsf{KL}}[\mathsf{p}\|\mathsf{q}]$	Kullback-Leibler divergence between probability distributions
$\mathtt{Dir}(\boldsymbol{\alpha})$	Dirichlet distribution with parameters $\boldsymbol{\alpha}$
$\mathtt{Cat}(\boldsymbol{\pi})$	Categorical distribution with probability vector $\boldsymbol{\pi}$
$\hat{\boldsymbol{\pi}}(\boldsymbol{x}) = f(\boldsymbol{x}; \boldsymbol{\theta})$	Predicted probabilities from model f
$\Gamma(x) = \int_0^\infty t^{x-1} e^{-t} dt$	The Gamma function
$\psi(x) = \frac{d}{dx} \log \Gamma(x)$	The Digamma function

Notations in Dialogue Belief/State Tracking

$\mathcal{O}$	Ontology of the dialogue system, representing all possible domain-slot pairs
d_m	`domain` m in the dialogue ontology
s_m	`domain-slot` pair m in the dialogue ontology
$\mathcal{V}_{\mathsf{s}_m}$	Set of plausible values for `domain-slot` pair m
t	Turn t in the dialogue
$v_t^{\mathsf{s}_m}$	Value of `domain-slot` pair s_m at turn t
$\mathcal{B}_t = \{(\mathsf{s}_m, v_t^{\mathsf{s}_m})\}_{\mathsf{s}_m \in \mathcal{O}}$	Dialogue state at turn t
$\boldsymbol{u}_t^{\mathsf{usr}}$	User utterance at turn t
$\boldsymbol{u}_t^{\mathsf{sys}}$	System utterance at turn t
a_t	System action (semantic) at turn t
o_t	User action (semantic) at turn t

Chapter 1

Introduction

1.1 Overview

Uncertainty is an inevitable aspect of our daily lives. We are constantly faced with a multitude of unpredictable situations and must take decisions based on incomplete and/or ambiguous information. The ability to recognise and handle uncertainty is a vital cognitive skill that plays a critical role in our decision-making processes (Stanovich, 2009, Chapter 6).

This ability allows us to take informed decisions, weighing the potential risks and rewards associated with different actions. Moreover, uncertainty serves as a driving force in our pursuit of knowledge. As we encounter the unknown, we selectively acquire new information, expanding our understanding and refining our mental models. This continuous process of learning from uncertainty not only enhances our ability to adapt and succeed, but also underpins the very essence of human curiosity.

Dialogue plays an integral role in our everyday interactions, where we skilfully navigate uncertainty to enable effective communication. By introducing topics that highlight the unknown or posing questions that challenge our assumptions, we encourage others to share their knowledge and engage in more meaningful conversations. Therefore, for software to engage effectively in dialogue with humans, it's crucial to recognise and manage uncertainty. By doing so, the system can determine the appropriate moments to ask questions, acquiring new knowledge and resolving ambiguities to optimise the efficiency of conversations. However, this is not reflected in current state-of-the-art conversational systems. Notably, models based on Large Language Models (LLMs), like ChatGPT, which are very large models trained to understand and generate human-like text, frequently display overconfidence in their responses, regardless of their actual accuracy. This overconfidence, as observed in studies by Chen and Mueller (2023) and Xiong et al. (2023), is a significant pain point, highlighting a discrepancy between human dialogue norms and the behaviour of advanced conversational agents.

In this thesis, we aim to explore the potential of leveraging uncertainty in deep learning models to enhance human-computer dialogue, with the aspiration of achieving conversation that closely resembles human-level interaction. To accomplish this, we introduce techniques for quantifying uncertainty in task-oriented dialogue systems. By integrating this uncertainty into the decision-making and the knowledge acquisition processes of our models, we strive to develop more effective and robust dialogue systems.

1.1.1 Task-oriented Dialogue Systems

Spoken dialogue systems are computer programs designed to engage in conversation with humans. Task-oriented dialogue systems, in particular, concentrate on providing users with information related to a specific goal, such as finding a restaurant or booking a flight. This differs from chat-based dialogue systems, which primarily aim to simulate casual social interactions. As we explore task-oriented dialogue systems in greater depth, it is important to recognise that some of the concepts and techniques discussed here may also be adapted for use in chat-based dialogue systems.

Task-oriented dialogue systems are typically modelled using a divide-and-conquer approach, breaking the problem into smaller, well-defined components. As depicted in Figure 1.1, a standard

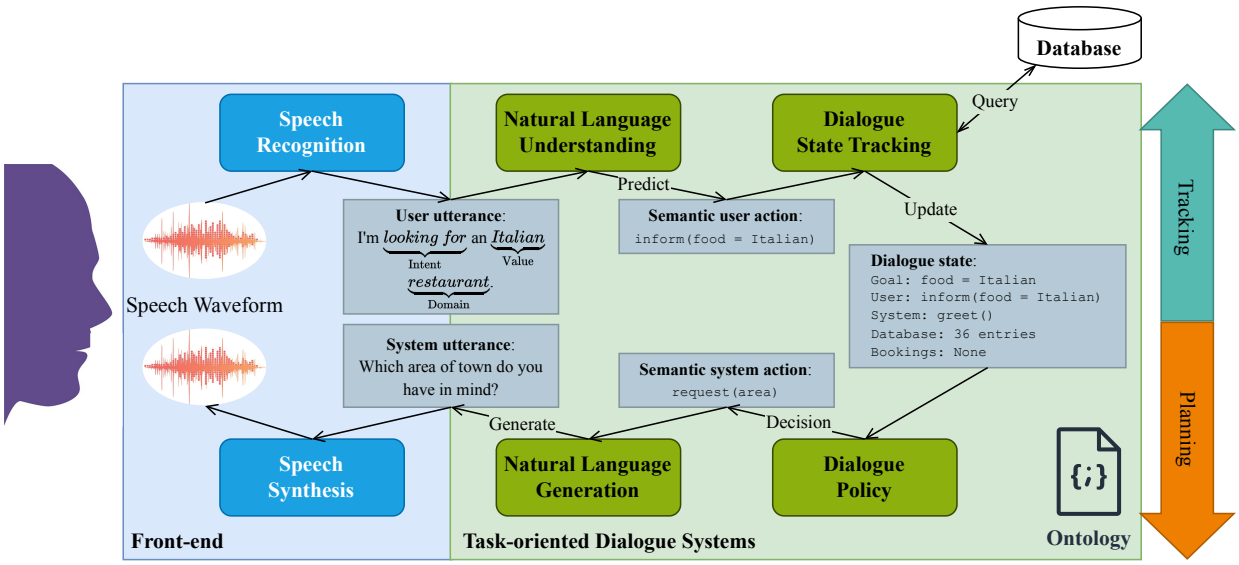


FIGURE 1.1: Key components of a modular task-oriented dialogue system. The speech recogniser converts the user's speech into text, which is then processed by the language understanding module to identify user acts. Following this, the state tracker extracts essential information from the user actions, the previous dialogue state and the last system action and modifies the user's goal. Using this updated goal, the state tracker queries an external database for relevant entities and generates the updated dialogue state. Using this state, the policy determines the optimal next action, and the language generation module generates the text to be synthesised as a response.

pipeline of modules is used to facilitate interactions with the dialogue system. At the front-end, modules (in blue) convert speech to text and vice versa. Thanks to large amounts of speech transcript data and recent advancements in speech models (Radford et al., 2023; Xiong et al., 2017), these modules perform with low error rates under known conditions. Consequently, we focus on text-based dialogue systems (in green) in this work.

Task-oriented dialogue systems are typically built upon an underlying ontology. This ontology serves as a structured representation of all potential topics that may arise during a dialogue. It often includes various domains (e.g., restaurants), categories of interest, also known as slots (such as price range, area, and cuisine), and their corresponding values (e.g., inexpensive, city centre, Italian). Furthermore, the ontology encompasses potential user or system intents (e.g., request, inform, confirm). In addition, the system is linked to a database that contains all possible entities and their attributes, including, for instance, an exhaustive list of restaurants, each detailed by location, types of food served, price range, contact information, and more.

In a modular dialogue system, such as the one depicted in Figure 1.1, the utterance of the user is processed by the natural language understanding (NLU) module, which is tasked with extracting/identifying the user's intents and actions. For example, when given the text "*I'm looking for an Italian restaurant.*", the NLU would predict the action `inform(food = Italian)`.

The dialogue state tracker (DST) keeps track of the state of the dialogue. This state, at a minimum, contains the latest user and system actions, the goal of the user, results of a database lookup, and any information related to bookings made during the dialogue, as seen in Figure 1.1. The goal of the user is typically represented as the most likely combination of domains, slots and values describing user needs. An alternative to DST is the dialogue belief tracker (DBT). The DBT is a model, which represents the users goal as a distribution of all possible goals defined in the ontology. Figure 1.2 depicts the difference between DST and DBT components.

Based on the dialogue state, the dialogue policy determines the optimal system actions to effectively respond to the user's query or statement. For instance, if the user is inquiring about restaurants, the policy might employ an action such as `request(area)`, prompting the user to specify the desired

location for restaurant recommendations. This action is subsequently processed by a natural language generation (NLG) model, which converts it into coherent and human-like text, for example "*Which area of town do you have in mind?*". Finally, the text is synthesised into speech, providing an audible response to the user (illustrated in Figure 1.1).

In modular task-oriented dialogue systems, the selected response is exclusively dependent on the state predicted by the DST/DBT modules (Young et al., 2010). The accuracy of this prediction, consequently, is crucial for the success of the dialogue, as its rooted in the systems capability to accurately interpret and conclude the users stated intentions and provided information. Hence, this work primarily focuses on the dialogue state/belief tracking and natural language understanding components of task-oriented dialogue systems. In the subsequent section, we will delve into the challenges associated with dialogue state/belief tracking.

1.2 Challenges in Dialogue Belief Tracking

Dialogue is the process of interacting using natural language, and forms the core of human-computer interaction, yet ensuring accurate understanding and tracking of these conversations remains a complex challenge. The subsequent sections detail these challenges, emphasising the importance of addressing them for improved communication between humans and machines.

Firstly, there is the issue of inaccurate understanding. In approximately every 2-3 turns in a dialogue, the understanding and state tracking models fails to fully understand the user utterances (Heck et al., 2020b; Kim et al., 2020; Zhang et al., 2020). This frequently results in a discrepancy between what the user intends and how the system perceives it.

Secondly, data constraints come into play. Even with the existence of comprehensive datasets like MultiWOZ (Budzianowski et al., 2018; Eric et al., 2020; Han et al., 2021; Ye et al., 2022; Zang et al., 2020) and the Schema-guided dialogue dataset (SGD) (Rastogi et al., 2020), there is a significant problem with data sparsity (a situation where datasets, although large, lack diversity and richness in content). The intricate ontologies underlying these datasets, consists of a vast array of domains, slots, and values, which give rise to multi-domain dialogues that are challenging to navigate. However, there is a stark contrast in diversity between the MultiWOZ and SGD datasets, SGD contains considerably more domains for conversation, in fact, over double compared to MultiWOZ, and around $10,000$ more unique values, but the number of unique tokens (individual units of meaning, such as words or phrases) in SGD exceeds that of MultiWOZ by merely $7,000$. This demonstrates the limited diversity in utterances within the SGD dataset despite its broader range of domains. Furthermore, even the largests of these datasets contains fewer than $20,000$ dialogues, with less than $500,000$ turns, as depicted in Table 1.1. In contrast, datasets available for other tasks such as machine translation are much larger. For instance, the German to English WMT17 dataset contains over $5,000,000$ German to English sentence pairs, which include more than $30,000$ unique tokens (Bojar et al., 2017).

Dataset	Number of dialogues	Average number of turns per dialogue	Average number of tokens per turn	Number of Unique tokens	Number of Domains	Number of Slots	Number of Values
MultiWOZ	8438	13.46	**13.13**	23689	7	24	4510
SGD	**16142**	**20.44**	9.75	**30352**	**16**	**214**	**14139**

TABLE 1.1: Comparison of the MultiWOZ and Schema guided dialogue (SGD) datasets.

The tracking task is made exceptionally challenging by the sheer number of domains, slots, and values in their ontologies. This is evidenced by the relatively low accuracies achieved on these datasets. State-of-the-art dialogue state tracking (DST) performance has stagnated at an accuracy of approximately 60% on these datasets (Heck et al., 2020a, 2022; Li et al., 2020). Many models compound this problem by having full confidence in these incorrect predictions, highlighting the importance of uncertainty estimation in dialogue state tracking. This leads us to the third challenge.

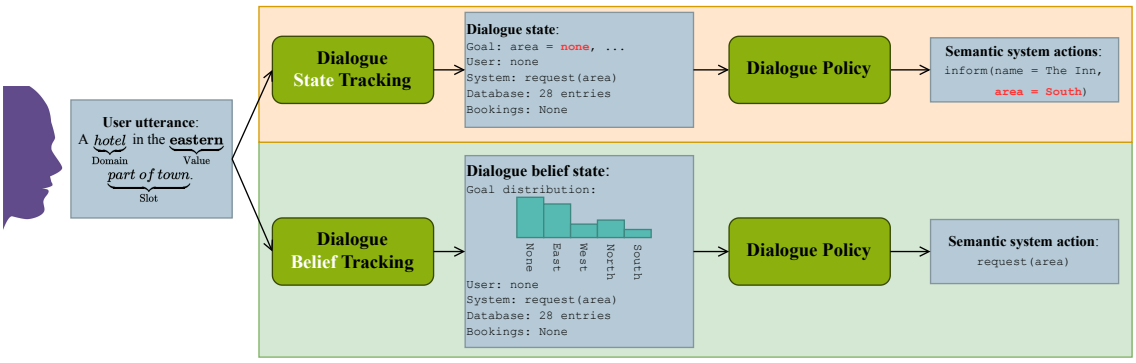


FIGURE 1.2: Error propagation poses a significant challenge in dialogue systems, necessitating mechanisms for estimating and managing uncertainty to rectify misunderstandings. This figure illustrates a comparison between two dialogue systems. The first system employs a dialogue **state** tracker (represented in orange) that disregards uncertainty information, offering only a single candidate dialogue state to the policy module. Consequently, this system cannot recover from misunderstandings, leading to inaccurate or irrelevant responses and a unsatisfactory user experience. Conversely, the second system incorporates a dialogue **belief** tracker (depicted in green), equipped with calibrated confidence estimates that encompass uncertainty information. Unlike its counterpart, the **belief** tracker presents a distribution of possible states to the policy module, reflecting the system's uncertainty about various aspects of the user's input, such as the specified hotel area. By considering this uncertainty, the policy module can infer that there was a misunderstanding regarding the hotel area and can opt to ask for confirmation. This allows the system to recover from potential misunderstandings and provide a better user experience.

The third challenge is the overconfidence exhibited by dialogue state tracking models. These models, on many occasions, exhibit high confidence in their predictions, even when incorrect. This overconfidence can mislead subsequent modules in the dialogue system, leading to what is also referred to as the *broken telephone* problem, where miss-information is transferred from one module to the next, causing the error to compound. This compounded error is termed as error propagation. Accurate uncertainty estimation can enhance system decisions and potentially help in recovering from misunderstandings. It can also improve user experiences by allowing the system to ask clarifying questions or undertake actions that mitigate the risk of failure due to such misunderstandings. Figure 1.2 provides an illustrative example of a system recovering from a belief tracker's error.

The fourth challenge is adaptability. Given the dynamic nature of human conversation, it is crucial for dialogue systems to be adaptable. Yet, a significant portion of existing models demonstrates inflexibility when introduced to unfamiliar domains, which constrains their applicability in real-world settings (Ren et al., 2018). Dialogue state/belief tracking models which rely on a fixed ontology demonstrate strong performance on datasets where the ontology contains a small number of domains and slots (Henderson et al., 2014b; Mrkšić et al., 2017), but struggle with more complex ontologies. Promising models that demonstrate generalisability to complex ontologies include discriminative models based on similarity matching (Lee et al., 2019; Ren et al., 2018; van Niekerk et al., 2021; Zhang et al., 2020), span prediction models (Heck et al., 2022, 2020b; Zhang et al., 2020), or generative language models for state tracking (Kim et al., 2020; Lin et al., 2021a; Lin et al., 2021b).

Lastly, the quality of training data is a concern. Training datasets often contain inconsistent and/or inaccurate labels. Such noise in the data not only pose challenges during the training process but might also introduce biases within the system. Dialogue datasets such as MultiWOZ contain significant noise, evident from the large number of annotation correction in the different versions of the dataset (Budzianowski et al., 2018; Eric et al., 2020; Han et al., 2021; Ye et al., 2022; Zang et al., 2020).

Striving towards more natural human-computer dialogue necessitates addressing these challenges.

Subsequent sections will explore these issues in greater depth, seeking to elucidate them and suggest viable solutions.

1.2.1 Calibration and Uncertainty

In the context of deep learning, *calibration* refers to the degree of alignment between a model's predicted probabilities and the actual observed outcomes. A well-calibrated model should ideally match its predicted probabilities with the observed empirical likelihood of a given dataset (Desai and Durrett, 2020).

However, there is a widespread understanding that deep learning models (complex function approximation algorithms inspired by the structure and function of the brain) optimised using maximum likelihood tend to produce overconfident, and hence poorly calibrated, predictions (Gal and Ghahramani, 2016). In addition to the optimisation objective, another contributing factor to this issue is the models inherent limitation in estimating knowledge uncertainty. Knowledge uncertainty, uncertainty arising from the models awareness of its own limitations and gaps in understanding intricate or unforeseen scenarios, is pivotal in enhancing the calibration of predictive outputs. Since deep learning models are often not equipped to gauge this form of uncertainty, they lack an intrinsic self-evaluation mechanism to moderate their confidence levels. This absence of self-regulation and introspection results in predictions that are not only overconfident but also potentially misleading. Acknowledging and integrating knowledge uncertainty can facilitate more balanced, reliable, and trustworthy model predictions, ensuring a closer alignment between the model's expressed confidence and its actual predictive accuracy. Techniques such as Bayesian deep learning (Blundell et al., 2015), dropout as a Bayesian approximation (Gal and Ghahramani, 2016), or ensembles (Lakshminarayanan et al., 2017) can be used to estimate knowledge uncertainty.

Drawing inspiration from psychological research, it is evident that recognising and managing uncertainty plays a vital role in human decision-making (Hirsh et al., 2012). The distinction between knowledge and data uncertainty (uncertainty arising from the variability and noise inherent in the data generation process) parallels findings from psychological studies that point to ambiguity and the occurrence of complex or unexpected events as sources of uncertainty (Bland and Schaefer, 2012). Addressing both data and knowledge uncertainty is crucial for dialogue belief tracking models to achieve better calibration. By improving calibration, dialogue systems can enhance their decision-making capabilities.

1.3 Contributions

In this thesis, key challenges in the field of dialogue systems are addressed, with a focus on dialogue belief tracking and the handling of uncertainty. The contributions can be summarised as follows:

1. **Enhanced performance and calibration for multi-domain dialogue belief trackers**: We apply various calibration techniques to a baseline dialogue belief tracker and demonstrate that a label-smoothed trained ensemble provides state-of-the-art calibration of belief state distributions and the highest accuracy among available belief trackers. Our proposed model achieves state-of-the-art performance in estimating well calibrated uncertainty scores. The proposed calibration methods can be applied to any neural dialogue belief tracking method, highlighting their broad applicability.

2. **Examining the downstream impact of uncertainty measures on policy optimisation**: We propose the use of total and knowledge uncertainties along with confidence scores to form a dialogue belief state. We further introduce SetSUMBT, a model capable of producing such belief states and knowledge uncertainty via distillation of ensembles. This tracker is able to estimate active domains and user requests in a dialogue, enabling more robust downstream

dialogue policy performance. Interactions with both simulated and real users confirm that these uncertainty metrics lead to more robust dialogue policy models.

3. **A semi-supervised active learning framework for sequential multi-output label problems with self-supervision and label validation**: We propose CAMELL, an active learning approach combining self-supervision and human-supervision to reduce the number of labels required for solving sequential multi-output label problems. Our method includes a label validation component to reject untrustworthy human annotations, enabling automated dataset annotation correction. In the context of dialogue belief tracking and machine translation, our approach outperforms strong baselines in robustness and data-efficiency. Experiments confirm that our label correction improves annotation quality. It demonstrates the applicability of the proposed framework to various sequential multi-output label problems, and reveals that dataset deficiencies can be addressed in a data-driven manner, potentially circumventing manual or rule-based annotation validation.

These contributions advance the estimation, management and utilisation of uncertainty inside a dialogue system and open the door to applications from wider domains. Moreover, they can increase system robustness to misunderstandings and ambiguities by employing uncertainty in the decision-making process, paving the way for effective human-computer interactions. Drawing inspiration from psychological findings on human uncertainty management (Hirsh et al., 2012), our work emphasises the importance of integrating these insights into machine learning models.

1.4 Thesis Structure

This thesis consists of seven chapters:

Chapter 1 presents a general introduction, setting the context and outlining the thesis objectives.

Chapter 2 offers an overview of deep learning, explaining its fundamental concepts, techniques, and applications.

Chapter 3 offers an overview of uncertainty estimation in deep learning, explaining its fundamental concepts, techniques, and applications.

Chapter 4 covers the necessary background on dialogue belief tracking, discussing its significance and the current state of research in the field.

Chapter 5 delves into uncertainty estimation and calibration in dialogue belief tracking and proposes a well-calibrated dialogue belief tracking model.

Chapter 6 introduces the SetSUMBT model (**Set** Similarity based **S**lot **U**tterance **M**atching **B**elief **T**racker), demonstrating how uncertainty can enhance the performance of the downstream dialogue policy model.

Chapter 7 presents the fundamentals of active learning and label validation, introducing CAMELL, our **C**onfidence-based **A**cquisition **M**odel for **E**fficient self-supervised active **L**earning with **L**abel validation.

Chapter 8 summarises the thesis's key findings, examines the implications of the presented research, and suggests directions for future investigation.

Chapter 2

Deep Learning Preliminaries

2.1 Overview

Deep learning, a branch of machine learning, has significantly impacted various domains ranging from computer vision to natural language processing. At the heart of deep learning is the deep neural network (DNN), a sophisticated framework designed for function approximation. Inspired by biological neural networks, DNNs consist of a hierarchical arrangement of interconnected neurons, which contribute to the learning and recognition of intricate patterns within data. These networks map input variables, $\boldsymbol{x}$, to their respective outcomes, $\boldsymbol{y}$, through a series of computations and activation functions.

The primary aim of this chapter is to lay down the foundational deep learning concepts. We will begin by delving into fully connected deep neural networks, discussing their architecture, and focusing on the role of activation functions (Section 2.2). We'll then explore the nuances of parameter optimisation, specifically emphasising objective functions and optimisation algorithms that refine the models (Section 2.3).

Sequential data processing using neural network architectures will be our next focus. We will introduce and explore Convolutional Neural Networks (CNNs) (Section 2.5), Recurrent Neural Networks (RNNs) (Section 2.6) and Transformers (Section 2.10), each serving specialised roles in handling different types of sequential data. To round off this chapter, we will explore the concept of language models (Sections 2.11, 2.12 and 2.13), outlining their tasks and utility in the broader context of machine learning and artificial intelligence.

By the end of this chapter, the reader will be equipped with the foundational knowledge required to understand the applications outlined in this thesis.

2.2 Fundamentals of Neural Networks

We begin our discussion by laying the foundation with the fundamentals of neural networks, which serve as the cornerstone of deep learning. Neural networks can be viewed as function approximators, denoted by $f_{\mathsf{NN}} : \mathcal{X} \to \mathcal{Y}$, that maps from the input space $\mathcal{X}$ to the output space $\mathcal{Y}$.

The most basic and prevalent architecture in the realm of neural networks is the fully connected neural network, often referred to as a feed forward neural network. This network architecture is distinguished by its interconnected neurons or nodes. Each neuron in a layer takes a weighted sum of its inputs and applies an activation function to this sum. The mathematical expression that describes this operation for a fully connected layer is:

$$f_{\mathsf{FC}}\left(\boldsymbol{x};\boldsymbol{\theta}\right) = \phi\left(\boldsymbol{x}\boldsymbol{W} + \boldsymbol{b}\right), \tag{2.1}$$

where $\boldsymbol{x} \in \mathcal{X}$ represents the input features, ϕ is the activation function, $\boldsymbol{W}$ is the weight matrix, and $\boldsymbol{b}$ is the bias vector. Here, $\boldsymbol{\theta}$ encompasses all the learnable parameters of the layer, which includes both the weight matrix and the bias vector.

The predicted outcome $\hat{\boldsymbol{y}}$ for a given input $\boldsymbol{x}$ is obtained as $\hat{\boldsymbol{y}} = f_{\mathsf{FC}}\left(\boldsymbol{x};\boldsymbol{\theta}\right)$.

2.2.1 Deep Neural Networks

We expand on basic neural networks to introduce deep neural networks (DNNs), distinguished by their multiple layers. In a fully connected DNN, each layer $g^{(i)}$, is a fully connected layer as defined in Equation 2.1. Considering a DNN with $m > 1$ layers and given a set of input features $x \in \mathbb{R}^{d_0}$, the fully connected DNN (illustrated in Figure 2.1) can be formalised as:

$$f_{\mathsf{FC}}(\boldsymbol{x};\boldsymbol{\theta}) = g^{(m-1)} \circ g^{(m-2)} \circ \cdots \circ g^{(1)} \circ g^{(0)}(\boldsymbol{x};\boldsymbol{\theta}),$$

with $\boldsymbol{\theta}$ representing the collective set of learnable parameters and $f \circ g(x) = f(g(x))$ representing a composition of functions. In DNNs, layers 0 to $m-2$ are typically referred to as "hidden" layers and layer $m-1$ as the output layer (as seen in Figure 2.1).

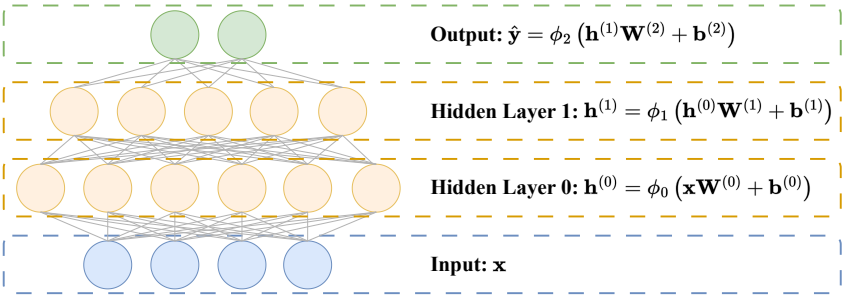


FIGURE 2.1: A prototypical deep neural network featuring two hidden layers.

Having established a foundational understanding of deep neural networks and their basic structure, a key component integral to their functionality is the activation function, ϕ. Activation functions infuse the necessary non-linearities into the model, enabling it to capture complex relationships in data. Let us delve deeper into the specifics of these functions.

2.2.2 Activation Functions

Activation functions, denoted as $\phi(\cdot)$, introduce non-linearities into the output of a neuron or a layer of neurons. Such non-linear transformations are important in deep learning models, especially given that without these non-linear transformations, a network, regardless of its depth, would be equivalent to a single linear layer and only capable of representing linear transformations (Goodfellow et al., 2016, Chapter 6). This would be restrictive, especially since real-world data often embodies complex, non-linear relationships between inputs and outputs. Over time, several activation functions have been proposed, with the sigmoid (Rumelhart and McClelland, 1987), softmax (Narayan, 1997), Rectified Linear Unit (ReLU) (Nair and Hinton, 2010), and hyperbolic tangent (`tanh`) (Lecun et al., 1998) functions being among the most widely used.

The Sigmoid function is characterized by its S-shaped curve and maps real-valued numbers to the $(0, 1)$ range. Mathematically, the sigmoid function is represented as:

$$\phi_{\mathsf{Sigmoid}}(z) = \frac{1}{1 + e^{-z}}$$

For substantial negative values of z, the sigmoid's output is approximately 0, whereas for large positive values, it nears 1. This bounded nature renders the sigmoid function especially suitable for generating probability outputs for binary classification tasks.

The Softmax function can be thought of as a multivariate extension of the sigmoid function. When dealing with multi-class classification tasks, the softmax ensures that the sum of all class probabilities equals 1. It is defined as:

$$\phi_{\texttt{Softmax}}(\boldsymbol{z}) = \frac{e^{\boldsymbol{z}}}{\sum_{j=1}^{d} e^{z_j}},$$

where $\boldsymbol{z}$ is a d-dimensional input vector.

The ReLU (Rectified Linear Unit) has gained popularity due to its simplicity and its ability to promote faster convergence during the training of deep networks. It is defined as:

$$\phi_{\texttt{ReLU}}(z) = \max(0, z)$$

The ReLU function activates a neuron if its input is positive; otherwise, it outputs zero.

The hyperbolic tangent (tanh) function outputs values in the range $(-1, 1)$. This property aids in efficient learning and provides a measure of control against the exploding gradient phenomenon. This phenomenon occurs when gradients become too large during parameter optimisation, leading to numerical instability and poor model performance. It is mathematically represented as:

$$\phi_{\texttt{tanh}}(z) = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}}$$

The selection of an appropriate activation function is often problem-dependent. For instance, ReLU and tanh are commonly employed within the hidden layers of a neural network. This is because of their roles in promoting efficient learning and in curbing the issues of vanishing or exploding gradients during parameter optimisation. Conversely, the sigmoid and softmax functions are frequently reserved for the output layer, converting the DNN's real-valued output into interpretable probability distributions (Szandała, 2021).

While the choice of activation function is crucial, ensuring the correct values for the network's parameters is equally paramount. A network with poorly optimised parameters, regardless of its structure and activation functions, would be inefficient. This leads us to the significant aspect of parameter optimisation in deep learning.

2.3 Parameter Optimisation

Optimisation is key to effective machine learning. This section lays out the techniques for optimising the parameters of neural networks. Given a training dataset $\mathcal{D} = \{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=1}^{N}$ consisting of N input-target pairs. Training a deep learning model essentially involves adjusting its parameters, denoted by $\boldsymbol{\theta}$, to minimise (or sometimes maximise) an objective function, represented as $\mathcal{L}(\mathcal{D}; \boldsymbol{\theta})$. The objective function, commonly referred to as the loss or cost function, serves to quantify the deviation between the model's predicted outcomes and the actual observed values. Central to this optimisation process is the employment of an iterative algorithm known as gradient descent.

In each iteration of the optimisation process, the model's parameters are updated in a specific direction that minimises the objective function. This direction is determined by the gradient of the objective function with respect to the parameters. The term *gradient* essentially refers to a vector of partial derivatives of the objective function, and it points in the direction of the steepest ascent of the function. To minimise the function, the parameters are moved in the opposite direction of the gradient. The extent to which the parameters are modified in this direction is controlled by a scalar factor known as the learning rate, denoted by η. In mathematical terms, a single update step in the gradient descent algorithm can be formally represented as:

$$\boldsymbol{\theta}^{\mathtt{new}} = \boldsymbol{\theta}^{\mathtt{old}} - \eta \nabla_{\boldsymbol{\theta}} \mathcal{L}\left(\mathcal{D}; \boldsymbol{\theta}^{\mathtt{old}}\right) \tag{2.2}$$

This equation captures the essence of iterative optimisation in deep learning: repetitively adjusting our model's parameters in the direction that reduces the error, as quantified by our objective function.

One notable variation of gradient descent is the use of mini-batches, aptly termed mini-batch gradient descent. Instead of using the entire dataset $\mathcal{D}$ to calculate the gradient, a small, random subset of the data is selected to approximate the gradient. The size of this mini-batch is a hyper-parameter, typically ranging from tens to hundreds. Mathematically, given a mini-batch $\mathcal{D}_{\mathtt{batch}} \subseteq \mathcal{D}$, the parameter update becomes:

$$\boldsymbol{\theta}^{\mathtt{new}} = \boldsymbol{\theta}^{\mathtt{old}} - \eta \nabla_{\boldsymbol{\theta}} \mathcal{L}\left(\mathcal{D}_{\mathtt{batch}}; \boldsymbol{\theta}^{\mathtt{old}}\right)$$

Mini-batch optimisation brings several advantages. It is computationally more efficient than batch gradient descent, offers faster convergence in wall-clock time, and can make effective use of hardware such as GPUs. The stochastic nature of mini-batch selection also introduces variability into the optimisation process. This randomness can help the model escape local minima in non-convex loss landscapes, making it potentially more robust in finding better overall solutions.

Having discussed parameter optimisation, we now turn our attention to the different types of objective functions that can be employed in the training process.

2.3.1 Objective Functions

The objective function, commonly referred to as the loss or cost function, quantifies the discrepancy between the predictions of the model and the actual outcomes. The choice of the objective function is pivotal and varies based on the nature of the task at hand. For instance, regression tasks frequently employ the mean-squared error (MSE) loss, whereas classification tasks often utilise the cross-entropy loss.

Mean-Squared Error (MSE): This loss function measures the average squared difference between the model predictions and the actual outcomes. In simple terms, it gauges the model's accuracy in terms of how close its numeric predictions are to the ground truth values. Mathematically, the MSE loss is given by:

$$\mathcal{L}_{\mathtt{MSE}}\left(\mathcal{D}; \boldsymbol{\theta}\right) = \frac{1}{N} \sum_{i=1}^{N} \left(y_i - f(\boldsymbol{x}_i; \boldsymbol{\theta})\right)^2$$

where $f(\boldsymbol{x}_i; \boldsymbol{\theta})$ is the model's prediction for the i-th observation.

Cross-entropy Loss: This loss function gauges the model's ability to assign correct probability scores to different classes in a classification problem. In essence, it quantifies how well the model's estimated probabilities align with the ground truth labels. A lower cross-entropy value indicates better alignment between predictions and actual outcomes. The mathematical representation of the cross-entropy loss is:

$$\begin{aligned}\mathcal{L}_{\mathtt{CE}}\left(\mathcal{D}; \boldsymbol{\theta}\right) &= -\log\left(\prod_{i=1}^{N} \mathsf{p}\left(\mathsf{Y} = y_i | \boldsymbol{x}_i, \boldsymbol{\theta}\right)\right) \\ &= -\sum_{i=1}^{N} \log \mathsf{p}\left(\mathsf{Y} = y_i | \boldsymbol{x}_i, \boldsymbol{\theta}\right)\end{aligned}$$

In this equation, $\mathsf{p}\left(\mathsf{Y} = y_i | \boldsymbol{x}_i, \boldsymbol{\theta}\right)$ represents the probability of the i-th data point belonging to the correct class.

Having established the role of the objective function in training deep learning models, our next step is to explore the optimisation algorithms employed to find the best possible model parameters, $\boldsymbol{\theta}$, that minimise the corresponding optimisation function.

2.3.2 Optimisation Algorithms

While the standard gradient descent optimisation, as described by Equation 2.2, has been foundational, it possesses inherent limitations. Namely, it can become stuck in local optima, preventing it from pursuing the global optimum, and its dependency on a singular learning rate for all parameters can impede efficient convergence. Modern optimisation techniques such as Momentum (Qian, 1999), Root Mean Square Propagation (RMSProp) (Hinton et al., 2012), and Adam (Kingma and Ba, 2017) have been introduced to address these challenges.

Momentum: Expanding on the principle of gradient descent, the Momentum method utilises a history of past gradients to hasten convergence. The term *momentum* refers to the moving average of the gradients. This technique assists in bypassing potential pitfalls like local minima and facilitates smoother transitions through saddle points (Qian, 1999). The parameter update rule, given the gradients $\boldsymbol{g}_t = \nabla_{\boldsymbol{\theta}} \mathcal{L}\left(\mathcal{D}; \boldsymbol{\theta}_{t-1}\right)$ is:

$$\begin{aligned} \text{Momentum: } \boldsymbol{m}_t &= \beta_1 \boldsymbol{m}_{t-1} + (1-\beta_1)\boldsymbol{g}_t \\ \text{Update Rule: } \boldsymbol{\theta}_t &= \boldsymbol{\theta}_{t-1} - \eta \boldsymbol{m}_t. \end{aligned} \tag{2.3}$$

Here β_1 is a hyper-parameters dictating the influence of the gradient in the moving average.

RMSProp (Root Mean Square Propagation): sets itself apart from traditional optimisation techniques by assigning distinct learning rates to each parameter, which enhances the efficiency of the training process. The uniqueness of this method lies in adjusting the global learning rate for each parameter based on the magnitude of its gradient. This individualised approach facilitates balanced and efficient parameter updates, overcoming the limitations of a uniform learning rate. In RMSProp, a parameter with a large gradient has a smaller learning rate to avoid large updates that can lead to instability or oscillation in the learning process. On the contrary, a parameter with a small gradient is assigned a higher learning rate, ensuring it can still be updated significantly to contribute effectively to the model's learning. This dynamic adjustment of learning rates according to gradient magnitudes ensures that each parameter is updated optimally, promoting faster and more stable convergence towards the minimum of the loss function (Hinton et al., 2012). The parameter update rule is:

$$\begin{aligned} \text{Magnitude: } \boldsymbol{\nu}_t &= \beta_2 \boldsymbol{\nu}_{t-1} + (1-\beta_2)\boldsymbol{g}_t^2 \\ \text{Update Rule: } \boldsymbol{\theta}_t &= \boldsymbol{\theta}_{t-1} - \eta \frac{\boldsymbol{g}_t}{\sqrt{\boldsymbol{\nu}_t} + \epsilon}. \end{aligned} \tag{2.4}$$

Here, β_2 is a hyper-parameter controlling the influence of the gradient magnitude $\boldsymbol{g}_t^2$.

Adam (Adaptive Moment Estimation): The Adam optimiser incorporates the best of both worlds: the momentum of gradient descent and the adaptive learning rate from RMSProp (Kingma and Ba, 2017). Notably, the Adam optimisation algorithm addresses an inherent bias in the initial updates. This bias emerges due to the initialisation of the moving averages $\boldsymbol{m}_t$ (first moment) and $\boldsymbol{\nu}_t$ (second moment) as vectors of zeros. The expanded moving average of gradients is:

$$\begin{aligned}
\boldsymbol{m}_t &= \beta_1 \boldsymbol{m}_{t-1} + (1-\beta_1)\boldsymbol{g}_t \\
&= \beta_1 \left(\beta_1 \boldsymbol{m}_{t-2} + (1-\beta_1)\boldsymbol{g}_{t-1}\right) + (1-\beta_1)\boldsymbol{g}_t \\
&= \beta_1^2 \left(\beta_1 \boldsymbol{m}_{t-3} + (1-\beta_1)\boldsymbol{g}_{t-2}\right) + \beta_1(1-\beta_1)\boldsymbol{g}_{t-1} + (1-\beta_1)\boldsymbol{g}_t \\
&= \beta_1^t \boldsymbol{m}_0 + (1-\beta_1)\sum_{i=0}^{t-1} \beta_1^i \boldsymbol{g}_{t-i}.
\end{aligned}$$

As a result:

$$\mathbb{E}_{\vec{\mathsf{G}}}\left[\vec{\mathsf{M}}_t\right] = \beta_1^t \boldsymbol{m}_0 + (1-\beta_1)\sum_{i=0}^{t-1} \beta_1^i \mathbb{E}\left[\vec{\mathsf{G}}_{t-i}\right].$$

Here, $\vec{\mathsf{M}}_t$ is the estimator for the gradient $\vec{\mathsf{G}}_t$, with both being vectors of random variables. Under the assumption that $\boldsymbol{m}_0 = \mathbf{0}$ and $\vec{\mathsf{G}}_t$ is stationary:

$$\begin{aligned}
\mathbb{E}_{\vec{\mathsf{G}}_t}\left[\vec{\mathsf{M}}_t\right] &= (1-\beta_1)\sum_{i=0}^{t-1} \beta_1^i \mathbb{E}\left[\vec{\mathsf{G}}_t\right] \\
&= (1-\beta_1^t)\mathbb{E}\left[\vec{\mathsf{G}}_t\right] \qquad \{\text{Geometric sum}\}.
\end{aligned}$$

Therefore:

$$\begin{aligned}
\mathtt{Bias}\left(\vec{\mathsf{M}}_t\right) &= (1-\beta_1^t)\mathbb{E}\left[\vec{\mathsf{G}}_t\right] - \mathbb{E}\left[\vec{\mathsf{G}}_t\right] \\
&= -\beta_1^t \mathbb{E}\left[\vec{\mathsf{G}}_t\right].
\end{aligned}$$

This indicates that $\vec{\mathsf{M}}_t$ is as a biased estimator of the gradient. Particularly, when t is small and β_1 is close to 1, $\boldsymbol{m}_t$ tends to be close to $\mathbf{0}$, which can undesirably influence the early gradient updates. To counteract this, Adam incorporates a bias-correction mechanism:

$$\hat{\boldsymbol{m}}_t = \frac{\boldsymbol{m}_t}{1-\beta_1^t} \tag{2.5}$$

Therefore $\mathbb{E}\left[\vec{\hat{\mathsf{M}}}_t\right] = \mathbb{E}\left[\vec{\mathsf{G}}_t\right]$ and $\mathtt{Bias}\left(\vec{\hat{\mathsf{M}}}_t\right) = \mathbf{0}$. With this correction, $\vec{\hat{\mathsf{M}}}_t$ is an unbiased estimator of the gradient. Hence, for smaller values of t and β_1 close to 1, $\hat{\boldsymbol{m}}_t$ is approximately $\boldsymbol{g}_t$. As t grows, $\hat{\boldsymbol{m}}_t$ tends towards $\boldsymbol{m}_t$, ensuring that the most recent gradients have a more substantial influence in the early updates. Similarly, Adam applies a correction for the learning rate adaptation based on the second moment. By combining momentum (as seen in Equation 2.3), gradient adjustment (refer to Equation 2.4), and bias correction (detailed in Equation 2.5), we derive the Adam update rule:

$$\begin{aligned}
\text{Momentum: } \hat{\boldsymbol{m}}_t &= \frac{\boldsymbol{m}_t}{1-\beta_1^t} \\
\text{Magnitude: } \hat{\boldsymbol{v}}_t &= \frac{\boldsymbol{v}_t}{1-\beta_2^t} \\
\text{Update Rule: } \boldsymbol{\theta}_t &= \boldsymbol{\theta}_{t-1} - \eta \frac{\hat{\boldsymbol{m}}_t}{\sqrt{\hat{\boldsymbol{v}}_t} + \epsilon}.
\end{aligned} \tag{2.6}$$

AdamW: AdamW is an extension of the Adam optimiser, specifically addressing its inefficiencies in weight regularisation (a penalty on the loss function to reduce overfitting by constraining model weights). The L2 regularisation loss is defined as:

$$\mathcal{L}_{\mathsf{L2}}\left(\mathcal{D}; \boldsymbol{\theta}_{t-1}\right) = \mathcal{L}\left(\mathcal{D}; \boldsymbol{\theta}_{t-1}\right) + \frac{\lambda}{2}\sum_{j=1}^{J} \theta_{t-1,j}^2$$

Therefore the new gradient is given by:

$$g_t = \nabla_\theta \mathcal{L}\left(\mathcal{D}; \boldsymbol{\theta}_{t-1}\right) + \lambda \boldsymbol{\theta}_{t-1}.$$

As a result, the regularisation term, $\lambda \boldsymbol{\theta}_{t-1}$, gets accumulated into both the momentum and adaptive gradient calculations. Given that the momentum term is effectively divided by the adaptive learning rate during the update (Equation 2.6), this leads to the neutralisation of the regularisation term. Consequently, weight regularisation in Adam often fails to be effective (Loshchilov and Hutter, 2019).

To rectify this, AdamW decouples the regularisation from the gradient. Instead of incorporating it into the gradient, the regularising term is directly incorporated into the weight update, ensuring proper and effective regularisation:

$$\text{Update Rule: } \boldsymbol{\theta}_t = \boldsymbol{\theta}_{t-1} - \eta \lambda \boldsymbol{\theta}_{t-1} - \eta \frac{\hat{\boldsymbol{m}}_t}{\sqrt{\hat{\boldsymbol{v}}_t} + \epsilon}.$$

Having examined neural networks, including their activation functions, objective functions, and optimisation techniques, we now turn our attention to those specifically crafted for processing sequential data. These networks are instrumental in natural language processing tasks, notably in applications like dialogue state tracking.

2.4 Sequential Models

While DNNs can be applied across various scenarios, certain challenges, especially in domains like natural language processing and dialogue systems, demand consideration of the temporal sequence of data. Sequential models are precisely designed to address such needs. They make predictions based on a series of inputs rather than isolated individual samples. When presented with a sequence $\boldsymbol{x}_{1:t} = \langle \boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_t \rangle$, the model, denoted as $f\left(\boldsymbol{x}_{1:t}; \boldsymbol{\theta}\right)$, aims to predict the outcome y_t. A well known sequential DNN is the convolutional neural network (CNN).

2.5 Convolutional Neural Networks

Convolutional Neural Networks (CNNs) are specialised neural networks tailored for processing structured data. Originating from the study of the animal visual cortex, CNNs have been instrumental in achieving state-of-the-art performance in tasks like image classification and speech recognition.

The cornerstone of CNNs is the *convolutional* operation. Unlike fully connected networks which might process an entire image or sequence as a single entity, the convolutional operation processes data region-by-region, identifying local patterns. This is done using a filter (or kernel), which slides across the input data, applying the same learned weights to each region and producing feature maps (as illustrated in Figure 2.2). These feature maps, consisting of region specific features, retain spatial hierarchies, making CNNs adept at recognising patterns regardless of their position. As a result, CNNs possess a unique property: translation invariance. This allows them to recognise patterns regardless of their position in the input.

Consider a $1-$dimensional CNN layer, denoted as $f_{\mathsf{CNN}} : \mathbb{R}^{t \times d_0} \to \mathbb{R}^{t-l+1 \times d_0}$. This layer uses a filter, consisting of learnable parameters $\boldsymbol{F} = \begin{bmatrix} \boldsymbol{f}_1 & \boldsymbol{f}_2 & \cdots & \boldsymbol{f}_l \end{bmatrix}^\intercal \in \mathbb{R}^{l \times d_0}$, which can be thought of as a small window of length l. Figure 2.2 provides a visual representation of this convolution process.

Given an input sequence $\boldsymbol{X} \in \mathbb{R}^{T \times d_0}$ with length T and d_0 features, the *convolutional* operation is defined as:

$$\boldsymbol{h}_t = \sum_{j=1}^{l} \boldsymbol{x}_{t+j-1} \odot \boldsymbol{f}_j \quad \text{for } t = 1, 2, .., T - l + 1,$$

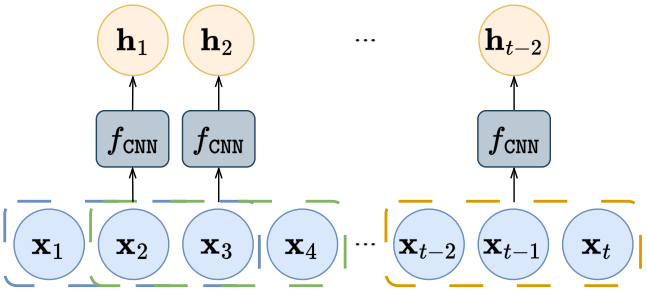


FIGURE 2.2: Schematic illustration of the convolution operation within a 1D CNN layer. In this visualisation, a sequence of input data points, ranging from $\boldsymbol{x}_1$ to $\boldsymbol{x}_t$, is processed by a convolutional filter of width 3. Each group of three adjacent data points (depicted by the blue, green, and orange outlines) forms a window to which the filter is applied. As the filter slides over the input sequence one step at a time, it computes a feature vector $\boldsymbol{h}_i$ for each window, resulting in a transformed set of feature vectors capturing the essential patterns and characteristics inherent in each window of the input sequence.

where $\boldsymbol{x}_i$ and $\boldsymbol{f}_i$ is the i-th rows of their respective matrices. The result is a feature matrix, $\boldsymbol{H} = \begin{bmatrix}\boldsymbol{h}_1 & \boldsymbol{h}_2 & \cdots & \boldsymbol{h}_{T-l+1}\end{bmatrix}^\intercal \in \mathbb{R}^{T-l+1\times d_0}$, which encapsulates the local patterns recognised by the filter. The dimensions of this output matrix depend on the filter's length, yielding an output length of $T-l+1$.

In mathematical terms, the CNN layer is given by:

$$f_{\text{CNN}}\left(\boldsymbol{X};\boldsymbol{F}\right) = \left[\sum_{j=1}^{l} \boldsymbol{x}_{t+j-1} \odot \boldsymbol{f}_j\right]_{t=1}^{T-l+1}.$$

Here $\odot$ symbolises element-wise multiplication. It is important to note that the dimensions of the output can change based on stride and padding. The stride is the number of positions the filter moves after each operation, while padding (often zeroes) is added to fill the spatial dimensions of the output. However, the above equations consider a stride of 1 and no padding.

Having discussed CNNs, which excel in spatial pattern recognition, let us move to Recurrent Neural Networks, adept at temporal pattern recognition.

2.6 Recurrent Neural Networks

Recurrent neural networks (RNNs) stand out in the vast landscape of neural networks due to their inherent ability to recognize and process the temporal nature of data. Unlike traditional DNNs, which treat each input independently, or CNNs, which treat neighbourhoods of inputs independently, RNNs establish connections across time steps, allowing them to learn from and reference previous instances in a sequence. This temporal *memory* capacity equips RNNs to excel in tasks that involve sequential data, making them invaluable for applications where understanding the past can influence accurate predictions about the future.

As visualised in Figure 2.3, a recurrent neural network comprises a deep neural network structure that is applied repetitively at each time step t. Taking the form of the function $f_{\text{RNN}}\left(\boldsymbol{x}_t, \boldsymbol{h}_{t-1}; \boldsymbol{\theta}\right)$, an RNN accepts the input features at the current time t, $\boldsymbol{x}_t$, and the hidden state from the previous time step, $\boldsymbol{h}_{t-1}$. These components, endowed with a set of adjustable parameters $\boldsymbol{\theta}$, permit the RNN to propagate context from past observations throughout a sequence. Consequently, the refreshed hidden state and output $\boldsymbol{y}_t$ at time t are defined as:

$$\boldsymbol{y}_t, \boldsymbol{h}_t = f_{\text{RNN}}\left(\boldsymbol{x}_t, \boldsymbol{h}_{t-1}; \boldsymbol{\theta}\right).$$

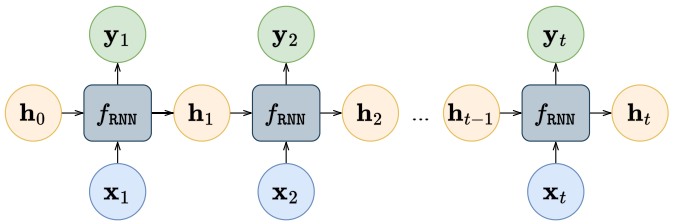


FIGURE 2.3: Illustration of a Recurrent Neural Network (RNN). RNNs are tailored to process sequences of data, combining inputs at each time step t with the hidden state from the preceding step to propagate information throughout the sequence. Subsequently, the model yields an updated hidden state and corresponding output.

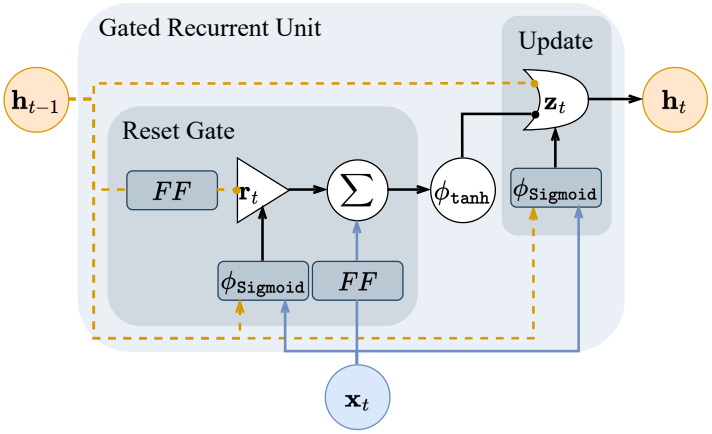


FIGURE 2.4: Schematic representation of the Gated Recurrent Unit (GRU). The symbols FF and ϕ_{Sigmoid} represent fully connected neural networks with linear and sigmoid activations, respectively. The variables $\boldsymbol{r}_t$ and $\boldsymbol{z}_t$ denote the gating mechanisms crucial for controlling information flow. In illustration, a triangle gate followed by a sum indicates a mechanism where only one input is weighted. In contrast, the half-moon shaped gate indicates a mechanism where both inputs are weighted. Circles within the schematic are indicative of auxiliary operations, including summation and the application of the tanh function.

When considering a Gated Recurrent Unit (GRU) (Cho et al., 2014) depicted in Figure 2.4 the recurrent function f_{RNN} is detailed as:

$$
\begin{aligned}
f_{\text{RNN}}\left(\boldsymbol{x}_t, \boldsymbol{h}_{t-1}; \boldsymbol{\theta}\right) &= (\mathbf{1} - \boldsymbol{z}_t) \odot \tilde{\boldsymbol{h}}_t + \boldsymbol{z}_t \odot \boldsymbol{h}_{t-1}, \text{ where} \\
\boldsymbol{z}_t &= \phi_{\texttt{Sigmoid}}\left(\boldsymbol{x}\boldsymbol{W}^{\texttt{xz}} + \boldsymbol{h}_{t-1}\boldsymbol{W}^{\texttt{hz}} + \boldsymbol{b}^{\texttt{z}}\right) \quad \{\text{Update gate}\}, \text{ and} \\
\tilde{\boldsymbol{h}}_t &= \phi_{\texttt{tanh}}\left(\boldsymbol{x}_t\boldsymbol{W}^{\texttt{xt}} + \boldsymbol{b}^{\texttt{xt}} + \boldsymbol{r}_t \odot \left(\boldsymbol{h}_{t-1}\boldsymbol{W}^{\texttt{ht}} + \boldsymbol{b}^{\texttt{ht}}\right)\right). \text{ Where} \\
\boldsymbol{r}_t &= \phi_{\texttt{Sigmoid}}\left(\boldsymbol{x}_t\boldsymbol{W}^{\texttt{xr}} + \boldsymbol{h}_{t-1}\boldsymbol{W}^{\texttt{hr}} + \boldsymbol{b}^{\texttt{r}}\right) \quad \{\text{Reset gate}\}.
\end{aligned}
$$

With $\boldsymbol{\theta}$ representing the assortment of the GRU's adjustable parameters. The GRU is characterised by two pivotal gates: the reset gate, which influences the quantity of historical data captured in the interim state, and the update gate, which steers the integration of past and current states.

To gain a better understanding of the use of RNNs in sequential modelling, we next introduce the RNN encoder-decoder model.

2.7 The RNN Encoder-Decoder Model

Consider the task of translating an input sequence $\boldsymbol{x} = \langle x_1, x_2, \cdots, x_{T_x} \rangle$ into a target sequence $\boldsymbol{y} = \langle y_1, y_2, \cdots, y_{T_y} \rangle$. One well known model for such a task is the RNN encoder-decoder model.

This model consists of two components: the encoder (illustrated in green in Figure 2.5), which transformers the input sequence into a fixed-length context vector, and the decoder (illustrated in red in Figure 2.5), tasked with translating this context vector into the desired target sequence.

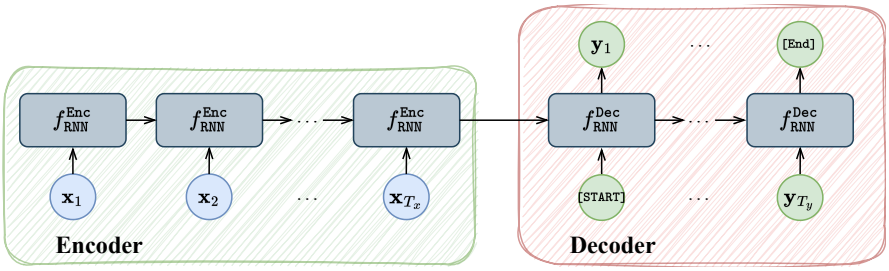


FIGURE 2.5: RNN Encoder-Decoder model: The encoder distils the input sequence into a context vector, while the decoder, conditioned on this context, generates the corresponding target sequence.

2.7.1 The Encoder

The encoder operates as a preliminary conversion module, processing the input sequence $\boldsymbol{x}_{1:T_x}$ and transforming it into a compressed, fixed-length context vector $\boldsymbol{h}^{\text{enc}}$. It can be mathematically described by:

$$\boldsymbol{h}^{\text{enc}} = f_{\text{Enc}}(\boldsymbol{x}) = f_{\text{RNN}}(\boldsymbol{x}).$$

This context vector serves as the initial state for the decoder and captures the essence of the input sequence.

2.7.2 The Decoder

The decoder aims to generate the target sequence based on the received context vector $\boldsymbol{h}^{\text{enc}}$. The decoder is defined as:

$$\boldsymbol{y} = f_{\text{Dec}}(\boldsymbol{y}_{\text{in}}, \boldsymbol{h}^{\text{enc}}) = f_{\text{RNN}}(\boldsymbol{y}_{\text{in}}, \boldsymbol{h}^{\text{enc}}).$$

The input to the decoder, denoted as $\boldsymbol{y}_{\text{in}}$, is a sequence starting with a special `[START]` element followed by elements of the target sequence, $y_{\text{in}} = \langle \texttt{[START]}, \boldsymbol{y}_1, \boldsymbol{y}_2, ..., \boldsymbol{y}_{T_y} \rangle$. The inclusion of the `[START]` element indicates the start of the sequence generation.

The decoder, operating in an auto-regressive manner, generates each token of the target sequence conditioned on the preceding tokens and the context vector. This ensures that the generation is contextually anchored. The generation ends when an `[END]` token is generated, marking the completion of the target sequence.

The RNN encoder-decoder model can be expressed as:

$$\boldsymbol{y} = f(\boldsymbol{x}, \boldsymbol{y}_{\text{in}}) = f_{\text{Dec}}(\boldsymbol{y}_{\text{in}}, f_{\text{Enc}}(\boldsymbol{x})).$$

However, such encoder-decoder models sometimes struggle to account for long-range dependencies, which is a critical factor in tasks requiring an understanding of the entire context. This sets the stage for a more specialised mechanism, the Attention Mechanism, that further improves the handling of sequential data, especially long-range dependencies within sequential data.

2.8 The Attention Mechanism

The attention mechanism enhances the capabilities of traditional architectures like RNNs and CNNs by offering a more context-aware understanding of data. It addresses the issue of long-range dependencies in sequences, enabling models to selectively focus on elements that are highly relevant to a given task (Bahdanau et al., 2015; Luong et al., 2015; Vaswani et al., 2017).

In the attention mechanism, the inputs are divided into three distinct components: queries, keys, and values. A query serves as a question, that the model poses, to identify relevant elements in the input sequence. Keys and values, on the other hand, encapsulate the information within the input/context data. Specifically, keys help determine which parts of the input the model should attend to, while values represent the content of these relevant parts.

To gather pertinent information, the model computes a weighted sum of the values. The weights, or "attention scores", are calculated through a compatibility function, denoted as ψ. Each attention score signifies the relevance of its corresponding value to the query. Formally:

$$g_{\mathtt{Att}}\left(\boldsymbol{V},\boldsymbol{K},\boldsymbol{Q}\right)=\psi\left(\boldsymbol{Q},\boldsymbol{K}\right)\boldsymbol{V}\in\mathbb{R}^{n_q\times d_v},$$

where matrices $\boldsymbol{Q}\in\mathbb{R}^{n_q\times d_k}$, $\boldsymbol{K}\in\mathbb{R}^{n_v\times d_k}$, and $\boldsymbol{V}\in\mathbb{R}^{n_v\times d_v}$ represent the queries, keys, and values. Furthermore, n_q represents the length of the query sequence, n_v denotes the length of the input sequence, d_k signifies the size of the query and key representations, and d_v indicates the size of the value representation. The scaled dot-product compatibility function is given by:

$$\psi_{\mathtt{SDP}}\left(\boldsymbol{Q},\boldsymbol{K}\right)=\phi_{\mathtt{Softmax}}\left(\frac{\boldsymbol{Q}\boldsymbol{K}^{\mathsf{T}}}{\sqrt{d_k}}\right)\in\mathbb{R}^{n_q\times n_v}.$$

This scaling ensures that the softmax activation's outputs remain within a suitable range. Assuming the vectors' elements in $\boldsymbol{Q}$ and $\boldsymbol{K}$ have expected value 0 and variance 1, the following relationships hold:

$$\begin{aligned}\boldsymbol{q}\cdot\boldsymbol{k}&=\sum_{j=1}^{d_k}q_ik_i,\ \text{therefore}\\ \mathbb{E}\left[\vec{\mathsf{Q}}\cdot\vec{\mathsf{K}}\right]&=\sum_{j=1}^{d_k}\mathbb{E}\left[\mathsf{Q}_i\right]\mathbb{E}\left[\mathsf{K}_i\right]=0,\ \text{and}\\ \mathtt{var}\left(\vec{\mathsf{Q}}\cdot\vec{\mathsf{K}}\right)&=\sum_{j=1}^{d_k}\left(\mathtt{var}(\mathsf{Q}_i)\mathtt{var}(\mathsf{K}_i)+\mathtt{var}(\mathsf{Q}_i)\mathbb{E}[\mathsf{K}_i]^2+\mathtt{var}(\mathsf{K}_i)\mathbb{E}[\mathsf{Q}_i]^2\right)=d_k.\end{aligned}$$

This indicates that the values of the dot-product typically fall in the range $\left(-3\sqrt{d_k},3\sqrt{d_k}\right)$ (the factor of 3 approximates a 99% confidence interval). However, as noted by Bengio et al. (2003), the softmax function performs optimally within the $(-3,3)$ range. Values that fall outside this interval result in softmax outputs of approximately 0 or 1, leading to slow and unstable training. The scaling by $\sqrt{d_k}$ ensures the softmax function's inputs lie typically in the range $(-3,3)$, aiding in faster and more stable training.

Now that we have grasped the concept of attention, let us see how it can be incorporated into RNNs for enhanced performance.

2.9 The RNN Encoder-Decoder with Attention

Building upon the RNN encoder-decoder model described in Section 2.7, we integrate the attention mechanism, detailed in Section 2.8, to improve the models ability in managing long-range dependencies. The integration of attention mechanism is instrumental, allowing the model to allocate focused attention on pertinent segments of the input sequence during the decoding phase, thus enabling a

more contextually enriched generation of the target sequence. Here the encoder produces a sequence of contextual vectors $\boldsymbol{H}^{\text{enc}}$:

$$\boldsymbol{H}^{\text{enc}} = f_{\text{Enc}}(\boldsymbol{x}) = f_{\text{RNN}}(\boldsymbol{x}).$$

The final element of this sequence serves as the initial state for decoder.

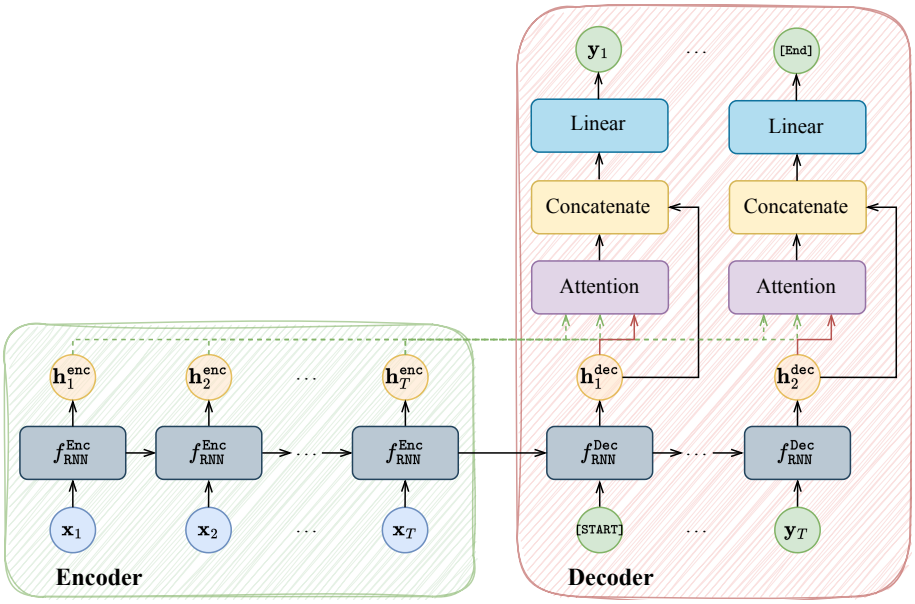


FIGURE 2.6: RNN Encoder-Decoder Featuring Attention.

2.9.1 The Decoder with Attention

While the foundational structure of the decoder remains the same as the one described in Section 2.7.2, the incorporation of the attention mechanism in an important improvement. The enhanced decoder is expressed as:

$$f_{\text{Dec}}(\boldsymbol{y}_{\text{in}}, \boldsymbol{H}^{\text{enc}}) = \left[g_{\text{att}}\left(\boldsymbol{H}^{\text{enc}}, \boldsymbol{H}^{\text{enc}}, \tilde{\boldsymbol{H}}\boldsymbol{W}^{\text{Q}}\right) \quad \tilde{\boldsymbol{H}}\right] \boldsymbol{W}^{\text{comb}},$$
$$\text{and } \tilde{\boldsymbol{H}} = f_{\text{RNN}}(\boldsymbol{y}_{\text{in}}, \boldsymbol{h}^{\text{enc}}_{T_x}).$$

In this equation, the attention mechanism, represented by g_{att}, calculates the weighted context from all encoder states $\boldsymbol{H}^{\text{enc}}$ and the current decoder state $\tilde{\boldsymbol{H}}\boldsymbol{W}^{\text{Q}}$. The $\boldsymbol{W}^{\text{Q}} \in \mathbb{R}^{d_{\text{model}} \times d_{\text{model}}}$ and $\boldsymbol{W}^{\text{comb}} \in \mathbb{R}^{2d_{\text{model}} \times d_{\text{model}}}$ are transformation matrices that adapt the features for the attention calculation and combine the attention context with the decoders features, respectively.

This combination of attention and an RNN encoder-decoder results in a model with a dynamic context for each element generated, rather than a static, fixed-length context vector. It provides a focused lens, enabling the model to attend to different portions of the input sequence as needed, enhancing its performance in complex sequence-to-sequence mapping tasks, particularly those involving long-range dependencies. Having combined attention with RNNs, we are now well-equipped to understand the Transformer model, which leverages attention in a unique way.

2.10 The Transformer

Building on our understanding of the sequential processing in Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs), we now transition to the Transformer model. Unlike RNNs, which process sequences iteratively, and CNNs that focus on local spatial hierarchies,

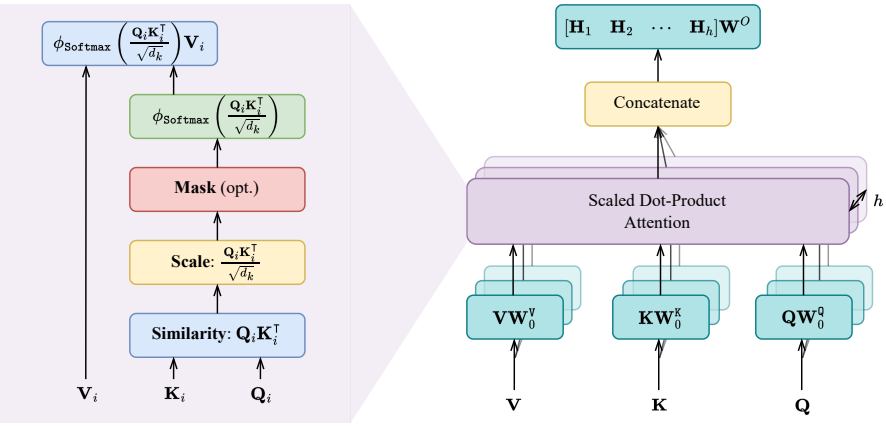


FIGURE 2.7: Illustration of single head and multi-head scaled dot-product attention. This attention mechanism calculates attention scores using dot-product similarity between query and key representations. Masking is applied when needed, as in masked language modelling or to prevent the decoder from peeking into the future. The multi-head attention comprises several individual attention mechanisms, each acting on a different projection of the input. The concatenated outcomes of these heads yield the final output representations.

Transformers employ a mechanism called self-attention (based on the attention introduced in Section 2.8). This mechanism allows them to assign varying degrees of importance to different parts of an input sequence, regardless of how far apart these parts are. This unique property effectively addresses the long-range dependency challenge that is sometimes a limiting factor in RNNs. Moreover, the parallel processing capability of Transformers gives them an edge in computational efficiency.

Delving deeper into its architecture, the Transformer model, introduced by Vaswani et al. (2017), operates on an encoder-decoder structure. The encoder ingests the input data, $\boldsymbol{x}_{1:T_x}$, converting it into hidden representations, $\boldsymbol{H}^{\text{enc}}$. Based on these representations, the decoder subsequently produces the output, $\boldsymbol{y}_{1:T_y}$.

Breaking down the Transformer model, we encounter several integral components:

Attention Mechanism: As the cornerstone of the Transformer, this mechanism allows the model to consider different parts of the input when processing a particular word or token.

Dense Feature Transformation: This component further processes the outputs from the attention mechanism, ensuring the extracted features are suitable for downstream tasks.

Positional Encodings: This layer is responsible for injecting positional information, important for processing sequential data.

In subsequent sections, each of these components will be explored in depth, providing a comprehensive understanding of their inner workings and significance in the Transformer's architecture.

2.10.1 Multi-head Attention

The multi-head attention mechanism (illustrated in Figure 2.8) comprises several attention *heads*, allowing the model to attend to various parts of the input differently for each head. These distinct attentions enrich the model's understanding of the data, which has proven instrumental in the Transformer's success in tasks like language modelling. The multi-head attention is formally defined as:

$$\begin{aligned} g_{\mathtt{MHA}}(\boldsymbol{V},\boldsymbol{K},\boldsymbol{Q}) &= \begin{bmatrix}\boldsymbol{H}^{(1)} & \boldsymbol{H}^{(2)} & \cdots & \boldsymbol{H}^{(h)}\end{bmatrix}\mathbf{W}^{\mathsf{O}} \in \mathbb{R}^{n_q \times d_v}, \\ \text{where } \boldsymbol{H}^{(i)} &= g_{\mathtt{Att}}\left(\boldsymbol{V}\boldsymbol{W}_i^{\mathsf{V}}, \boldsymbol{K}\boldsymbol{W}_i^{\mathsf{K}}, \boldsymbol{Q}\boldsymbol{W}_i^{\mathsf{Q}}\right) \in \mathbb{R}^{n_q \times d_v^h} \text{ for } i = 1, 2, ..., h. \end{aligned} \tag{2.7}$$

Here, the matrices $\mathbf{W}_i^{\mathsf{Q}} \in \mathbb{R}^{d_k \times d_k^h}$, $\mathbf{W}_i^{\mathsf{K}} \in \mathbb{R}^{d_k \times d_k^h}$, $\mathbf{W}_i^{\mathsf{V}} \in \mathbb{R}^{d_v \times d_v^h}$, and $\mathbf{W}^{\mathsf{O}} \in \mathbb{R}^{h d_v^h \times d_v}$ are learnable projection matrices. Employing multiple projections increases the models capacity, allowing it to capture a diverse and rich set of features from the input. This enhances the ability of the model to focus on varied sequence elements simultaneously, fostering a more comprehensive and robust understanding of data patterns.

2.10.2 Self-attention

In contrast to RNN and CNN models, the transformer does not process neighbourhoods of observations together, but rather considers the full context for each observation in the sequence. Self-attention is a variation of attention where the query, key and value are all set to the sequence of observations. This allows this mechanism to create rich contextual representations for each observation by focusing on the observations most relevant for understanding that specific observations. This means these mechanisms can easily deal with long term dependencies in sequential data. Mathematically the self-attention multi-head attention module is represented as:

$$g_{\mathtt{SA}}(\boldsymbol{X}) = g_{\mathtt{MHA}}(\boldsymbol{X}, \boldsymbol{X}, \boldsymbol{X}), \tag{2.8}$$

where $\boldsymbol{X}$ is a sequence of vectors.

2.10.3 Dense Feature Transformation Layers

Dense feature transformation layers, often simply called position-wise feed-forward layers in the context of the Transformer architecture, play a pivotal role in refining feature representations. Comprising two linear layers connected by a ReLU activation function, these layers are tasked with expanding and then contracting the feature space.

Let us delve into the mechanics. Given a feature transformation hidden size $d_{\mathtt{FT}}$, the transformation can be mathematically described as:

$$g_{\mathtt{FT}}(\boldsymbol{X}) = \phi_{\mathtt{ReLU}}\left(\boldsymbol{X}\boldsymbol{W}_1 + \boldsymbol{b}_1\right)\boldsymbol{W}_2 + \boldsymbol{b}_2, \tag{2.9}$$

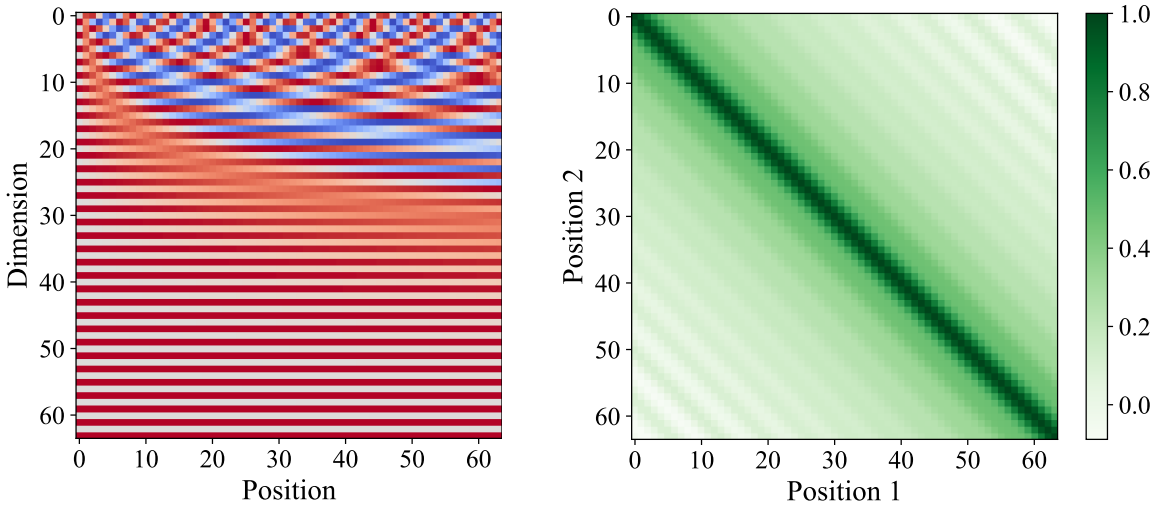
where

- $\boldsymbol{W}_1 \in \mathbb{R}^{d_{\mathtt{x}} \times d_{\mathtt{FT}}}$ and $\boldsymbol{b}_1 \in \mathbb{R}^{d_{\mathtt{FT}}}$ are the weight matrix and bias vector for the first layer.
- $\boldsymbol{W}_2 \in \mathbb{R}^{d_{\mathtt{FT}} \times d_{\mathtt{x}}}$ and $\boldsymbol{b}_2 \in \mathbb{R}^{d_{\mathtt{x}}}$ correspond to the second layer.

This large hidden layer gives the model high capacity for learning complex representations. Each neuron can learn to recognise different features or combinations of features from the input data. When we have many neurons, the layer can potentially learn a very rich and diverse set of features. Such feature transformation layers can be seen as a form of "memory" in a neural network allowing the model to remember information.

While these layers do not hold "memory" in a classical sense, they do facilitate the Transformer in storing representations or transformations of the input. This storage is implicit: the layer refines input features, emphasising certain elements, merging others, or even generating new feature combinations that encapsulate higher-order abstractions.

Empirical studies have shown the setting $d_{\mathtt{FT}} = 4d_{\mathtt{x}}$ is efficient (Devlin et al., 2019; Radford et al., 2018; Vaswani et al., 2017). This ratio strikes a balance, offering a comprehensive feature transformation capability without overfitting to the training dataset.



(A) Variation of positional encodings across different positions and dimensions, illustrating the geometric progression of wavelengths.

(B) Similarity heatmap showing the similarity of positional encodings for nearby sequence positions, and how this similarity diminishes as the distance between positions increases.

FIGURE 2.8: Visual representation of the sine-cosine positional encodings and their properties.

2.10.4 Positional Encodings

While the transformer architecture's parallel processing ability is a strength, it comes with an inherent limitation: the model lacks the ability to inherently discern the order or position of observations in a sequence. This is crucial, especially for tasks like language understanding, where word order often dictates meaning.

To address this, Vaswani et al. (2017) introduced the concept of positional encodings. These are added to the inputs, allowing the model to consider the position in a sequence. The proposed method uses sinusoidal functions of varying frequencies to compute the positional encodings. These positional encodings share the same dimension, d, as the input representations. For a token at position p the encoding is:

$$g_{\text{pos}}(p) = \begin{bmatrix} \sin(\lambda_0 p) & \cos(\lambda_0 p) & \sin(\lambda_1 p) & \cos(\lambda_1 p) & \cdots & \sin\left(\lambda_{\frac{d}{2}-1} p\right) & \cos\left(\lambda_{\frac{d}{2}-1} p\right) \end{bmatrix},$$

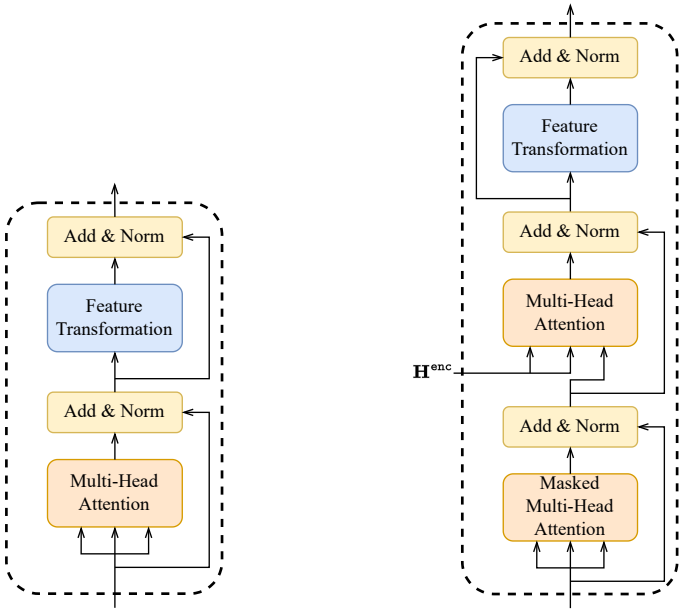
where $\lambda_m = 10000^{\frac{-2m}{d}}$.

For a sequence of inputs $\boldsymbol{x}$ of length T, the positional embedding function returns a matrix of positional embeddings:

$$g_{\text{pos}}(\boldsymbol{x}) = \begin{bmatrix} g_{\text{pos}}(0) & g_{\text{pos}}(1) & \cdots & g_{\text{pos}}(T-1) \end{bmatrix}^\intercal \in \mathbb{R}^{T \times d}.$$

The sine-cosine positional encodings exhibit three important properties, each contributing to the model's understanding of sequence position. Firstly, the encoding at position $t + k$ can be expressed as a linear combination of the encoding at position t (See Appendix Theorem 1). This property ensures that the change in positional encoding between any two positions in the sequence depends solely on the interval k between them. This leads to a consistent representation of relative positional differences, irrespective of the absolute location within the sequence.

Secondly, the distance between two observations in terms of the positions is captured. This is because the encodings are designed such that the similarity between the encodings diminishes as the distance between the corresponding observations increases (See Figure 2.8 (B) and Appendix Theorem 2).



(A) Transformer Encoder Layer (B) Transformer Decoder Layer

FIGURE 2.9: Transformer Encoder and Decoder Layers

Lastly, the wavelengths of the sine and cosine functions used in the positional encodings vary according to a geometric progression, ranging from 2π to $10000 \cdot 2\pi$ (See Appendix Theorem 3 and Figure 2.8 (A)). This scaling mechanism allows each dimension in the positional encoding to capture positional information at significantly different scales. Consequently, the model gains the capacity to understand a wide variety of positional dependencies and to develop a rich representation of element positions within a sequence.

2.10.5 The Transformer Layer

As illustrated in Figure 2.9 (A), a standard Transformer layer comprises two main components: a multi-head self-attention mechanism and a position-wise feature transformation layer. The self-attention mechanism is designed to recognise dependencies and relationships between tokens, thereby focusing on relevant parts of the input sequence. This mechanism is enhanced by a residual connection, which sums the input and output of the self-attention block. The residual connection allows the layer to learn what can be thought of as 'residuals' or valuable nuances in representation, aided by its inherent ability to approximate the identity function.

Layer normalisation is employed to maintain a consistent output distribution across various inputs, effectively stabilising the learning process. The second main component of the Transformer layer is the dense feature transformation layer (Section 2.10.3). A typical transformer layer is defined as:

$$f(\boldsymbol{X}) = \Phi\left(\tilde{\boldsymbol{H}} + g_{\mathsf{FT}}(\tilde{\boldsymbol{H}})\right), \quad \{\text{Equation 2.9}\} \text{ where}$$
$$\tilde{\boldsymbol{H}} = \Phi\left(\boldsymbol{X} + g_{\mathsf{SA}}(\boldsymbol{X})\right) \quad \{\text{Equation 2.8}\}.$$

The layer normalisation function, vital for maintaining consistency in output distribution across layers, is:

$$\Phi(\boldsymbol{x}) = \frac{\boldsymbol{x} - \mathbb{E}[\boldsymbol{x}]}{\sqrt{\mathsf{var}(\boldsymbol{x}) + \epsilon}} \odot \gamma + \beta, \tag{2.10}$$

where γ and β are learnable parameters and ϵ a small positive constant. This function normalises the output of the components of the layer such that:

$$\begin{aligned} \mathbb{E}\left[\Phi(\boldsymbol{x})\right] &= \boldsymbol{\beta} \\ \mathtt{var}\left(\Phi(\boldsymbol{x})\right) &= \gamma^2, \end{aligned} \tag{2.11}$$

Having explored the intricacies of the attention mechanism, encoder-decoder architecture, and transformer models, we now turn our attention to the domain of language modelling.

2.11 The Transformer Language Model

Introduced by Vaswani et al. (2017), transformer-based language models leverage attention mechanisms to concurrently process input sequences, a capability contrasting with the inherent sequential processing of RNNs. This allows transformers to efficiently capture long-range dependencies in text without the time-step limitations associated with RNNs.

2.11.1 Token Embeddings

The essence of natural language processing using deep learning models lies in the transformation of words or tokens into numerical representations, commonly referred to as embeddings. Transformer models employ a specialised tokenisation technique known as "word-piece tokenisation". Instead of representing each word as a whole, word-piece tokenisation breaks words into sub-tokens. This segmentation proves beneficial in several ways:

Shared Semantics: Common sub-tokens between words help the model understand shared semantics. For instance, the sub-token "un-" in both "unhappy" and "unfortunate" conveys the idea of negation.

Handling Rare Words: By decomposing rare or out-of-vocabulary words into familiar sub-tokens, the model can infer their meanings without having seen the entire word during training.

As a result, the vocabulary of the model consists of a set of tokens and the embedding layer of the model learns an embedding for tokens rather than words. The token embedding layer transforms each token in the input text into a continuous vector representation. This continuous form allows the model to capture semantic nuances between different tokens more effectively than discrete forms, such as integers or one-hot encoded vectors.

Mathematically, the operations performed by the token embedding layer can be formulated as:

$$g_{\mathtt{emb}}(\boldsymbol{x}) = \mathbb{1}(\boldsymbol{x})\boldsymbol{E},$$

where the element in the i-th row and j-th column of the one-hot matrix is defined as:

$$\mathbb{1}(\boldsymbol{x})_{i,j} = \begin{cases} 1 & \text{if } x_i = j \\ 0 & \text{otherwise} \end{cases}.$$

Here $i \in \{1, 2, \cdots, |\boldsymbol{x}|\}$ and $j \in \{0, 1, ..., \mathtt{vocab\_size} - 1\}$, and:

- $\boldsymbol{x}$ is an input vector containing the integer IDs that represent the tokens in the input text. Each integer ID x_i corresponds to a token in the model's vocabulary.
- $\mathbb{1}(\boldsymbol{x})$ denotes the one-hot encoded matrix of the input vector $\boldsymbol{x}$. Each row in this matrix corresponds to a token in the input and contains a single '1' at the column index that matches the token's integer ID, with all other entries being '0'.

- $\boldsymbol{E}$ is a learnable embedding matrix of dimensions $\texttt{vocab\_size} \times d_{\texttt{model}}$, where $\texttt{vocab\_size} \in \mathbb{N}^+$ is the size of the vocabulary and $d_{\texttt{model}}$ is the dimensionality of the embeddings. Each row in $\boldsymbol{E}$ represents the vector embedding of a unique token in the model's vocabulary.

By performing the matrix multiplication $\mathbb{1}(\boldsymbol{x})\boldsymbol{E}$, each row of one-hot encoded vectors picks out the corresponding token embedding from $\boldsymbol{E}$, resulting in a matrix $g_{\texttt{emb}}(\boldsymbol{x})$ where each row is the vector embedding of a token in the input text.

2.11.2 Transformer Encoder Layer

A transformer encoder layer can be mathematically represented as:

$$\begin{aligned} g_{\texttt{enc}}(\boldsymbol{X};\boldsymbol{\theta}) &= \Phi\left(\tilde{\boldsymbol{H}} + g_{\texttt{FT}}\left(\tilde{\boldsymbol{H}}\right)\right) \quad \{\text{Equation 2.9}\} \\ \tilde{\boldsymbol{H}} &= \Phi\left(\boldsymbol{X} + g_{\texttt{SA}}\left(\boldsymbol{X}\right)\right) \quad \{\text{Equation 2.8}\}, \end{aligned}$$

where $\boldsymbol{\theta}$ is the set of parameters for the self-attention and feature transformation layers.

2.11.3 Transformer Decoder Layer

Distinct from the encoder, the decoder layer, illustrated in Figure 2.9 (B), also incorporates encoder features, $\boldsymbol{H}^{\texttt{enc}}$:

$$g_{\texttt{dec}}(\boldsymbol{X}, \boldsymbol{H}^{\texttt{enc}};\boldsymbol{\theta}) = \Phi\left(\tilde{\boldsymbol{H}} + g_{\texttt{FT}}\left(\tilde{\boldsymbol{H}}\right)\right) \quad \{\text{Equation 2.9}\} \tag{2.12}$$

$$\tilde{\boldsymbol{H}} = \Phi\left(\hat{\boldsymbol{H}} + g_{\texttt{MHA}}\left(\boldsymbol{H}^{\texttt{enc}}, \boldsymbol{H}^{\texttt{enc}}, \hat{\boldsymbol{H}}\right)\right) \quad \{\text{Equation 2.7}\} \tag{2.13}$$

$$\hat{\boldsymbol{H}} = \Phi\left(\boldsymbol{X} + g_{\texttt{SA}}\left(\boldsymbol{X}\right)\right) \quad \{\text{Equation 2.8}\}. \tag{2.14}$$

The multi-head attention mechanism (Equation 2.13) dynamically aligns the decoder with relevant portions of the input text by computing weighted sums of all encoder states. It allows for the handling of long-range dependencies, ensuring that each generated word is influenced by a customised, dynamically-created context. Figure 2.10 shows an example of how the attention mechanism aligns the generated text with the source text, by *focusing* on the relevant words in the source text.

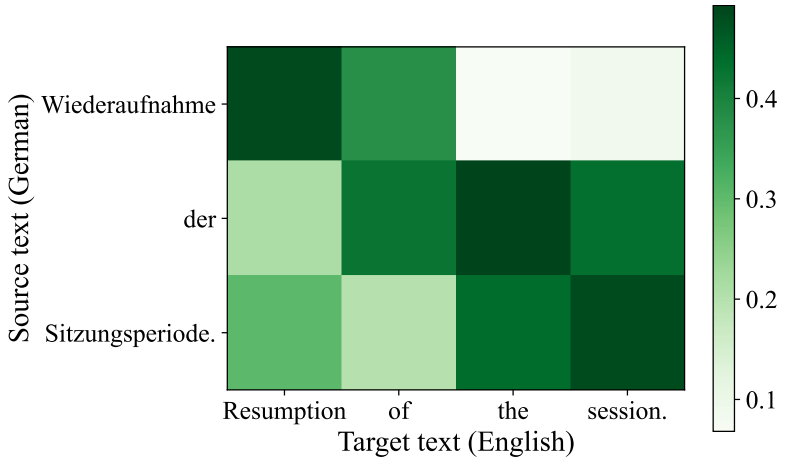


FIGURE 2.10: Illustration of Attention Mechanism in RNN Encoder-Decoder for WMT 17 DE-EN Translation Task. The figure displays the attention scores generated by the Transformer Model (Vaswani et al., 2017), showcasing how the attention mechanism aligns each target word (English) with the relevant source words (German). This visualisation provides insights into the model's ability to focus on relevant parts of the input sentence during translation.

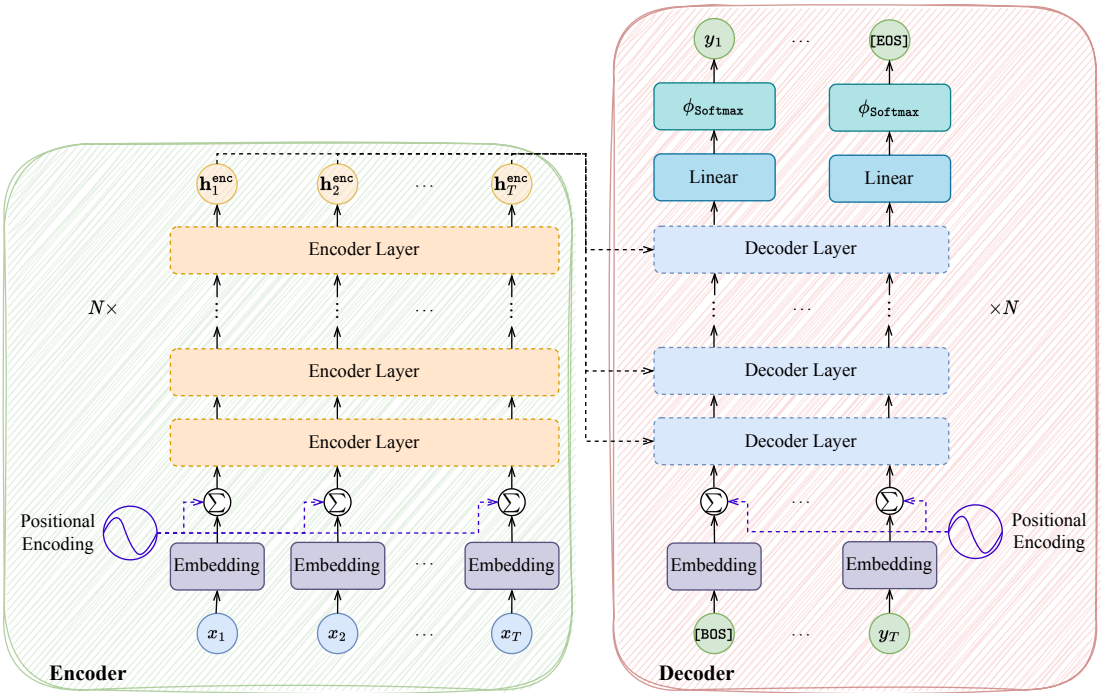


FIGURE 2.11: Transformer language model overview. It encompasses an encoder (in green) and a decoder (in red). The encoder utilises multi-head self-attention blocks to produce contextual representations of input tokens. The decoder, with its masked multi-head self-attention, ensures each output token considers all preceding tokens. This masking prevents the decoder from viewing future tokens during training, ensuring genuine auto-regressive generation. Additionally, the decoder employs encoder-decoder attention mechanisms and concludes with a classification layer for token generation.

2.11.4 The Model

The encoder model can be described as:

$$f_{\mathsf{Enc}}(\boldsymbol{x}) = g_{\mathsf{enc}}^{N} \circ g_{\mathsf{enc}}^{N-1} \circ \cdots \circ g_{\mathsf{enc}}^{1}(g_{\mathsf{emb}}(\boldsymbol{x}) + g_{\mathsf{pos}}(\boldsymbol{x})),$$

While, given $\boldsymbol{H}^{\mathsf{enc}} = f_{\mathsf{Enc}}(\boldsymbol{X})$, the decoder is:

$$f_{\mathsf{Dec}}(\boldsymbol{y}_{in}, \boldsymbol{H}^{\mathsf{enc}}) = \phi_{\mathtt{Softmax}}\left(\boldsymbol{H}^{\mathsf{dec}}\frac{\boldsymbol{E}^{\intercal}}{d}\right)$$
$$\boldsymbol{H}^{\mathsf{dec}} = g_{\mathsf{dec}}^{N} \circ g_{\mathsf{dec}}^{N-1} \circ \cdots \circ g_{\mathsf{dec}}^{1}(g_{\mathsf{emb}}(\boldsymbol{y}^{\mathsf{in}}) + g_{\mathsf{pos}}(\boldsymbol{y}^{\mathsf{in}}), \boldsymbol{H}^{\mathsf{enc}}),$$

where $\boldsymbol{E}$ is the same embedding matrix used to represent tokens in the input sequence. Crucially, both the encoder and decoder employ the same embedding matrix, $\boldsymbol{E}$, for token representation. This shared representation not only enforces semantic consistency but also serves as a regularisation technique. It curtails overfitting, speeds up convergence, and might enhance the model's overall performance.

This transformer language model (illustrated in Figure 2.11) can be formally defined as:

$$\mathsf{p}\left(\mathsf{Y}\middle|\boldsymbol{x}, \boldsymbol{y}^{\mathsf{in}}; \boldsymbol{\theta}\right) = f(\boldsymbol{x}, \boldsymbol{y}^{\mathsf{in}}; \boldsymbol{\theta}) = f_{\mathsf{Dec}}\left(\boldsymbol{y}^{\mathsf{in}}, f_{\mathsf{Enc}}(\boldsymbol{x}; \boldsymbol{\theta}); \boldsymbol{\theta}\right).$$

This model is trained on a dataset $\mathcal{D}$, consisting of input-target text pairs $\mathcal{D} = \{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=1}^{N}$. The learning objective is to minimise the cross-entropy loss between the predicted and target sequences. During training, the decoder input and target sequences are derived from the original target sequence $\boldsymbol{y}_i$. Specifically, the decoder inputs are constructed as $\boldsymbol{y}_i^{\mathsf{in}} = [\texttt{[BOS]}, y_{i,1}, y_{i,2}, \cdots, y_{i,T_i}]$ and the targets are $\boldsymbol{y}_i^{\mathsf{target}} = [y_{i,1}, y_{i,2}, \cdots, y_{i,T_i}, \texttt{[EOS]}]$. In this context, [BOS] and [EOS] denote 'Beginning Of

Sequence' and 'End Of Sequence' tokens, respectively, which are used to indicate the start and end of a sequence in the dataset. This objective is mathematically expressed as:

$$\mathcal{L}\left(\mathcal{D};\boldsymbol{\theta}\right) = -\frac{1}{N}\sum_{i=1}^{N}\frac{1}{T_i}\sum_{t=1}^{T_i}\log \mathsf{p}\left(\mathsf{Y} = y_{i,t}^{\mathtt{target}}\middle|\boldsymbol{x}_i, \boldsymbol{y}_{i,1:t}^{\mathtt{in}};\boldsymbol{\theta}\right).$$

During inference, each word in the output sequence is generated auto-regressively, optimising for the highest probability word at each step, until either a predetermined length, T, or an end-of-string token `[EOS]` is encountered. That is given that $y_0 = \texttt{[BOS]}$, the next word is generated as:

$$\hat{y}_t = \arg\max_{y} \mathsf{p}\left(\mathsf{Y} = y|\boldsymbol{x}, \boldsymbol{y}_{1:t-1};\boldsymbol{\theta}\right),$$

Such transformer language models could also comprise solely of an encoder or decoder.

2.12 Encoder Language Models

Encoder-only transformer language models utilise the encoder component of the Transformer (see Figure 2.12 (A)). A prominent representative of this design is BERT (Bidirectional Encoder Representations from Transformers) (Devlin et al., 2019). Its training procedure involves a masked language modelling objective, wherein random tokens are masked and the goal becomes predicting the original token based on the surrounding context. Distinctively, unlike the decoder, BERT integrates both preceding and subsequent contexts of a token, offering a comprehensive understanding of language structures.

The model can be represented as:

$$f(\boldsymbol{x};\boldsymbol{\theta}) = \phi_{\mathtt{Softmax}}\left(f_{\mathtt{Enc}}(\boldsymbol{x};\boldsymbol{\theta})\boldsymbol{W} + \boldsymbol{b}\right).$$

In encoder-only models, the training objective is to accurately predict masked tokens using the surrounding context. Given an input sequence $\boldsymbol{x}$ of length T, we derive $\boldsymbol{x}^{\mathtt{in}}$ by replacing random tokens with the `[MASK]` token. The desired output for this masked input is the original sequence, $\boldsymbol{x}$. To train this model, the cross-entropy loss is employed, formulated as

$$\mathcal{L}(\boldsymbol{x};\boldsymbol{\theta}) = -\sum_{t=1}^{T}\delta\left(\boldsymbol{x}_t^{\mathtt{in}}\right)\log \mathsf{p}\left(\mathsf{Y} = x_t|\boldsymbol{x}^{\mathtt{in}};\boldsymbol{\theta}\right).$$

Here

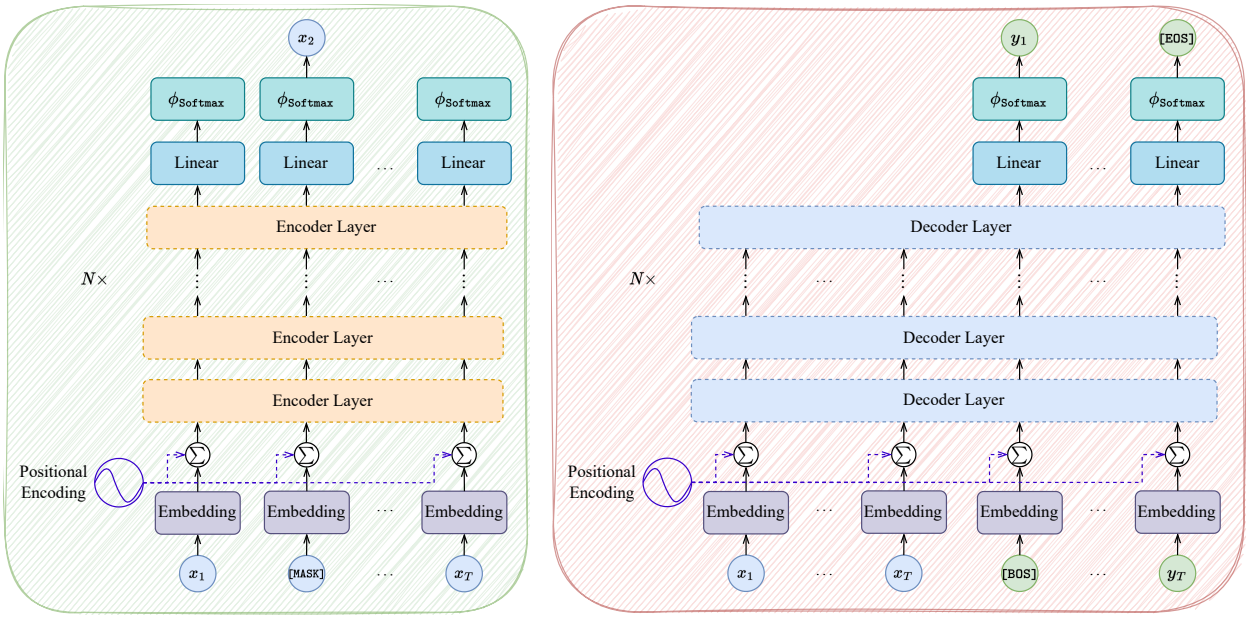
$$\delta\left(\boldsymbol{x}_t^{\mathtt{in}}\right) = \begin{cases} 1 & \text{if } \boldsymbol{x}_t^{\mathtt{in}} = [\mathtt{MASK}] \\ 0 & \text{otherwise} \end{cases},$$

such that the loss only includes predictions of masked tokens. Unlike auto-regressive models which aim to generate text, the primary goal of encoder-only models is to learn context-rich word embeddings. These embeddings can subsequently be used directly or fine-tuned for various downstream tasks, including dialogue state tracking.

2.13 Decoder Language Models

Decoder-only transformer language models exclusively harness the decoder section of the Transformer, aimed at generating new sequences (see Figure 2.12 (B)). Models under the GPT (Generative Pre-trained Transformer) umbrella exemplify this architecture (Brown et al., 2020; OpenAI, 2023; Ouyang et al., 2022; Radford et al., 2018, 2019).

In the GPT framework, the decoder's masked self-attention ensures that the prediction at position t is only influenced by positions $1, 2, \cdots, t-1$. This auto-regressive characteristic is vital for coherent



(A) Transformer Encoder (B) Transformer Decoder

FIGURE 2.12: Illustration of Encoder-Only and Decoder-Only Transformer Models. The encoder-only models, often referred to as masked language models, are trained to generate rich contextual representations for text tokens using the masked language modelling task. In this approach, certain tokens from the input sequence are masked, and the encoder then predicts these masked tokens based on the surrounding context. Conversely, decoder-only models engage in auto-regressive sequence generation. While generating text, the decoder's input incorporates both the input sequence and any previously generated tokens. This setup allows the model to attend to the entire input sequence during generation while masking future text tokens.

sequence generation. GPT's training regime maximises the likelihood of a token given its preceding tokens (objective is the same as for encoder-decoder language models), rendering it proficient at generating cohesive and contextually relevant text.

Though named differently, in these models, the decoder $f_{\texttt{Dec}}$ essentially functions as an encoder $f_{\texttt{Enc}}$ with provisions for future masking. Therefore, these models can be mathematically expressed as:

$$f(\boldsymbol{y}^{\texttt{in}};\boldsymbol{\theta}) = \phi_{\texttt{Softmax}}\left(f_{\texttt{Enc}}(\boldsymbol{y}^{\texttt{in}};\boldsymbol{\theta})\frac{\boldsymbol{E}^{\intercal}}{d}\right).$$

In summary, Transformer language models have catalysed advancements in NLP, both in terms of research and real-world applications. Encoder models such as BERT (with 110 million parameters for `BERT-Base`) excel in tasks necessitating comprehension of the input text, e.g., text classification and named entity recognition (Devlin et al., 2019; Liu et al., 2019). On the other hand, Encoder-Decoder and Decoder models like T5 (with up to 11 billion parameters for `T5-XXL`) and GPT (with 175 billion parameters for `GPT-3`), respectively, are highly effective for text generation tasks (Brown et al., 2020; Raffel et al., 2020).

2.14 Conclusion

The preliminary concepts and techniques in deep learning presented in this chapter, including neural networks, recurrent neural networks, convolutional neural networks, and the Transformer architecture, form the foundation for understanding and implementing more advanced models and applications, such as those discussed in Chapters 4-7.

Chapter 3

Uncertainty Estimation in Deep Learning

3.1 Overview

In Chapter 1, we highlighted the profound role that human cognition plays in managing and navigating uncertain terrains, and the foundational necessity for software to echo similar competencies. Yet, the challenge of uncertainty in deep learning stretches beyond merely achieving better human-computer interactions. Modern deep learning models, though sophisticated, are often found erring on the side of overconfidence. This overconfidence holds implications on the model's adaptability and relevance in diverse, real-world contexts (Grote and Berens, 2020).

While our understanding of human uncertainty, informed by psychological research, offers insights into the importance of uncertainty, there exists a pressing need to explore techniques that allow models to effectively quantify and integrate uncertainty. Such techniques not only promise to improve the precision and adaptability of our models but also pave the way for more effective, robust, and nuanced human-computer interactions (Collins et al., 2023).

In this chapter, we will delve into the intricacies of uncertainty estimation in deep learning, starting with the question: *What is uncertainty?*

3.2 What is Uncertainty?

Uncertainty, in the context of deep learning models, refers to the amount of doubt or lack of confidence a model has in its predictions (Gal, 2016). In essence, it captures the ambiguity and or volatility in the model's predictions occurring as a result of various internal and external factors.

Estimating uncertainty in deep learning models is important for various reasons. Firstly, it provides a deeper understanding of the predictions of a model by highlighting potential weaknesses or areas of low confidence. Secondly, in modular tasks such as medical diagnoses or autonomous vehicles, where predictions of a model are used in downstream decision making, unreliable predictions can lead to costly mistakes. Thirdly, since uncertainty provides us with an insight into the shortcomings of a model, uncertainty can aid informed strategies for model improvement, adaptive learning, and robust real-world deployment.

Importantly, a model capable of accurately estimating its own uncertainty is inherently more trustworthy. This allows users or downstream applications to gauge the reliability of the model's predictions, thereby minimising the risk of costly errors arising from incorrect or misleading outputs. A nuanced understanding of the types of uncertainty at play can further guide the proper utilisation and refinement of the model.

3.3 Types of Uncertainty in Deep Learning

Uncertainty in deep learning can be broadly categorised into two types: aleatoric (data-driven) and epistemic (knowledge-based) uncertainty (ibid.).

Aleatoric Uncertainty, also referred to as data uncertainty, refers to uncertainty that arises due to inherent variations or noise present in the data used for training a machine learning model. Generally, aleatoric uncertainty is considered irreducible; it will persist even if more data is gathered or a more complex model is used.

Epistemic Uncertainty, also known as knowledge uncertainty, refers to the uncertainty in a machine learning model's predictions which originates from its incomplete understanding, as a result of a lack of relevant data during training or the inherent complexity of the problem at hand. Unlike aleatoric uncertainty, epistemic uncertainty can be mitigated by gathering more data or employing more sophisticated models (Neal, 2012, Chapter 1).

Understanding these different types of uncertainty is the first step towards developing more reliable models. The next natural step is to explore how these uncertainties can be accurately estimated, which is the focus of the subsequent section.

3.4 Calibration Techniques

In the realm of machine learning, a model is deemed well-calibrated if the predicted confidences align with the empirical likelihood of those predictions. Notably, machine learning models, and particularly deep learning models, have a tendency to be overconfident in their inaccurate predictions, meaning they are not well-calibrated (Guo et al., 2017). To address this, this section outlines methods aimed for improving model calibration, i.e. better aligning the uncertainty of the model with the likelihood of its predictions. First, we investigate objective functions designed to encourage better-calibrated predictions. Subsequently, we delve into how ensemble methods can be employed not only to enhance calibration but also to provide estimates for both aleatoric and epistemic uncertainties (Lakshminarayanan et al., 2017). Finally, we explore the concept of ensemble distillation as a technique for achieving faster inference while maintaining the calibration (Hinton et al., 2015; Ryabinin et al., 2021).

3.4.1 Objective Functions

The choice of objective function plays a pivotal role in shaping the behaviour of a deep learning model, essentially guiding what the model learns from the training data. Therefore, objective functions serve not only as an optimisation tool but also as a mechanism for calibrating the model's predictive capabilities. Among the various options, the cross-entropy loss stands out for its ubiquity and efficacy. As previously introduced in Section 2.3.1, the cross-entropy loss can be mathematically defined for a training dataset $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{N}$, comprising N pairs of inputs and corresponding targets, as follows:

$$\begin{aligned}\mathcal{L}_{\mathsf{CE}}\left(\mathcal{D};\boldsymbol{\theta}\right) &= -\frac{1}{N}\sum_{i=1}^{N}\log \mathsf{p}\left(\mathsf{Y}=y_i|\boldsymbol{x}_i,\boldsymbol{\theta}\right)\\ &= -\frac{1}{N}\sum_{i=1}^{N}\sum_{c=1}^{K}\mathsf{p}\left(\mathsf{Y}=c|\boldsymbol{x}_i,y_i\right)\log \mathsf{p}\left(\mathsf{Y}=c|\boldsymbol{x}_i,\boldsymbol{\theta}\right),\end{aligned}$$

where K is the number of classes in the classification problem and $y_i \in [1, K]$. The conditional probability $\mathsf{p}\left(\mathsf{Y}=c|\boldsymbol{x}_i,y_i\right)$ is defined as:

$$\mathsf{p}\left(\mathsf{Y}=c|\boldsymbol{x}_i,y_i\right) = \begin{cases}1 & \text{if } y_i = c\\ 0 & \text{otherwise}\end{cases}. \tag{3.1}$$

Given this definition, the cross-entropy loss can also be expressed in terms of the Kullback-Leibler divergence (KL) as:

$$\begin{aligned}\mathcal{L}_{\mathsf{CE}}\left(\mathcal{D};\boldsymbol{\theta}\right) &= -\frac{1}{N}\sum_{i=1}^{N}\sum_{c=1}^{K}\mathsf{p}\left(\mathsf{Y}=c|\boldsymbol{x}_i,y_i\right)\log\left(\frac{\mathsf{p}\left(\mathsf{Y}=c|\boldsymbol{x}_i,\boldsymbol{\theta}\right)}{\mathsf{p}\left(\mathsf{Y}=c|\boldsymbol{x}_i,y_i\right)}\right)\\ &= \mathbb{E}_{\mathsf{P}\left(\vec{\mathsf{X}},\mathsf{Y}\middle|\mathcal{D}\right)}\left[D_{\mathsf{KL}}\left[\mathsf{p}\left(\mathsf{Y}|\vec{\mathsf{X}},\mathsf{Y}\right)\middle\|\mathsf{p}\left(\mathsf{Y}|\vec{\mathsf{X}},\boldsymbol{\theta}\right)\right]\right].\end{aligned}$$

The cross-entropy loss effectively encourages the model to approximate the true class distribution, $\mathsf{p}\left(\mathsf{Y}\middle|\vec{\mathsf{X}},\mathsf{Y}\right)$, which is a one-hot representation of the correct class (Goodfellow et al., 2016). However, this might not be ideal in cases where there is inherent uncertainty in the prediction. To address this, alternative loss functions like label-smoothing can be employed (Szegedy et al., 2016).

Before we introduce label-smoothing loss, we introduce the concept of *entropy*. The entropy of a random variable quantifies the average uncertainty or surprise associated with the outcomes that the variable can take. A higher entropy value indicates that the outcome is less predictable, leading to greater surprise when an outcome is revealed. On the other hand, a lower entropy implies that the outcomes are more predictable. Entropy is measured by taking a weighted sum of the log probabilities of all possible outcomes, scaled by their respective probabilities.

The label-smoothing objective function employs an adapted target distribution, $\mathsf{p}\left(\mathsf{Y}=c|\boldsymbol{x}_i,y_i\right)$ (as per Equation 3.1), to encourage the model to produce better-calibrated predictive distributions. In the label-smoothing objective, the target distribution is redefined as:

$$\mathsf{p}\left(\mathsf{Y}=c|\boldsymbol{x}_i,y_i;\epsilon\right)=\begin{cases}1-\epsilon & \text{if } y_i=c\\ \frac{\epsilon}{K-1} & \text{otherwise}\end{cases}.$$

where $\epsilon\in\left(0,\frac{K-1}{K}\right)$ serves as a hyper-parameter that controls the entropy level of the target distribution. Specifically, lower values of ϵ yield target distributions with minimal entropy, whereas higher ϵ values lead to distributions with increased entropy.

Having explored the role of loss functions in calibrating deep learning models, it is worth mentioning that an alternative yet complementary approach to model calibration involves the use of ensembles. Not only do ensembles offer a method for improving prediction accuracy, but they also provide valuable insights into model uncertainty, both aleatoric and epistemic (Malinin, 2019).

3.4.2 Ensembles

To estimate the likelihood of a given observation given a dataset $\mathcal{D}$, it is essential to integrate over all possible parameter values $\boldsymbol{\theta}$. Mathematically, this can be expressed as:

$$\begin{aligned}\mathsf{p}\left(\mathsf{Y}|\vec{\mathsf{X}},\mathcal{D}\right) &= \int\mathsf{P}\left(\mathsf{Y},\boldsymbol{\theta}|\vec{\mathsf{X}},\mathcal{D}\right)d\boldsymbol{\theta}\\ &= \int\mathsf{p}\left(\mathsf{Y}|\vec{\mathsf{X}},\boldsymbol{\theta}\right)\mathsf{P}\left(\boldsymbol{\theta}|\mathcal{D}\right)d\boldsymbol{\theta}\quad\{\text{Bayes's Theorem}\}\\ &= \mathbb{E}_{\mathsf{P}(\boldsymbol{\theta}|\mathcal{D})}\left[\mathsf{p}\left(\mathsf{Y}|\vec{\mathsf{X}},\boldsymbol{\theta}\right)\right].\end{aligned}\tag{3.2}$$

However, directly calculating this integral is often intractable, necessitating an approximation. One common approach involves sampling from the posterior distribution $\mathsf{p}\left(\boldsymbol{\theta}\middle|\mathcal{D}\right)$, also known as Markov Chain Monte Carlo (MCMC) estimation (Robert and Casella, 1999, Chapter 6):

$$\begin{aligned}&\mathsf{p}\left(\mathsf{Y}|\vec{\mathsf{X}},\mathcal{D}\right)\approx\frac{1}{M}\sum_{m=1}^{M}\mathsf{p}\left(\mathsf{Y}|\vec{\mathsf{X}},\boldsymbol{\theta}^{(m)}\right),\\ &\text{where }\boldsymbol{\theta}^{(m)}\sim\mathsf{P}\left(\boldsymbol{\theta}|\mathcal{D}\right).\end{aligned}\tag{3.3}$$

Ensemble methods offer a practical way to approximate this sampling. Each ensemble member represents a different sample from the posterior distribution, allowing us to approximate the true predictive distribution and thereby estimate total uncertainty:

$$\begin{aligned}\mathcal{H}\left(\mathsf{p}\left(\mathsf{Y}|\vec{\mathsf{X}},\mathcal{D}\right)\right) &= -\sum_{c=1}^{K}\mathsf{p}\left(\mathsf{Y}=c|\vec{\mathsf{X}},\mathcal{D}\right)\log\mathsf{p}\left(\mathsf{Y}=c|\vec{\mathsf{X}},\mathcal{D}\right)\\ &\approx -\frac{1}{M}\sum_{c=1}^{K}\sum_{m=1}^{M}\mathsf{p}\left(\mathsf{Y}=c|\vec{\mathsf{X}},\boldsymbol{\theta}^{(m)}\right)\log\left(\frac{1}{M}\sum_{m=1}^{M}\mathsf{p}\left(\mathsf{Y}=c|\vec{\mathsf{X}},\boldsymbol{\theta}^{(m)}\right)\right).\end{aligned}$$

Next, we introduce the concept of mutual information. In the context of information theory, mutual information serves as a quantitative measure of the information one random variable contains about another. Specifically, it gauges the extent to which knowing the value of one variable reduces uncertainty about the value of the other variable. The mutual information between random variables X and Y is defined as:

$$\mathcal{I}\left(\mathsf{X},\mathsf{Y}\right)=\mathbb{E}_{\mathsf{P}(\mathsf{X},\mathsf{Y})}\left[\log\left(\frac{\mathsf{P}\left(\mathsf{X},\mathsf{Y}\right)}{\mathsf{P}\left(\mathsf{X}\right)\mathsf{P}\left(\mathsf{Y}\right)}\right)\right].$$

In this equation, we calculate the expected value of the logarithm of the ratio between the joint probability distribution of X and Y and the product of their individual marginal distributions. When the two variables are independent, this ratio is 1, leading to a logarithm of zero and a mutual information of zero. This aligns with the intuitive understanding that if two variables are independent, knowledge of one provides no information about the other. Conversely, if the variables are dependent, the ratio will be greater than one, as the joint probability will exceed the product of the marginal probabilities. In such cases, the mutual information will be greater than zero, indicating that one variable does indeed provide information about the other.

When it comes to model parameters, denoted as $\boldsymbol{\theta}$, and outcomes, Y, a high mutual information means the outcome is very sensitive to these parameters. This might sound like a good thing, as it suggests the parameters and the outcome are closely linked. However, it also means uncertainty about the parameters can lead to inaccuracies in the predictions. This is why high mutual information in this context indicates a high level of *knowledge uncertainty* because small uncertainties in the parameters can lead to big uncertainties in the outcome. Additionally, high mutual information also means that knowing the outcome Y will provide a lot of information about what $\boldsymbol{\theta}$ should be. However, until Y is observed, this aspect contributes to *knowledge uncertainty*, the less we know about one, the less certain we are about the other (Malinin, 2019). In this context we can derive the knowledge uncertainty as:

$$\begin{aligned}\mathcal{I}\left(\mathsf{Y},\boldsymbol{\theta}|\vec{\mathsf{X}},\mathcal{D}\right) &= \int\sum_{k=1}^{K}\mathsf{P}\left(\mathsf{Y}=k,\boldsymbol{\theta}|\vec{\mathsf{X}},\mathcal{D}\right)\log\left(\frac{\mathsf{P}\left(\mathsf{Y}=k,\boldsymbol{\theta}|\vec{\mathsf{X}},\mathcal{D}\right)}{\mathsf{p}\left(\mathsf{Y}=k|\vec{\mathsf{X}},\mathcal{D}\right)\mathsf{P}\left(\boldsymbol{\theta}|\mathcal{D}\right)}\right)d\boldsymbol{\theta}\\ &= \int\mathsf{P}\left(\boldsymbol{\theta}|\mathcal{D}\right)\sum_{k=1}^{K}\mathsf{p}\left(\mathsf{Y}=k|\vec{\mathsf{X}},\boldsymbol{\theta}\right)\log\left(\frac{\mathsf{p}\left(\mathsf{Y}=k|\vec{\mathsf{X}},\boldsymbol{\theta}\right)\mathsf{P}\left(\boldsymbol{\theta}|\mathcal{D}\right)}{\mathsf{p}\left(\mathsf{Y}=k|\vec{\mathsf{X}},\mathcal{D}\right)\mathsf{P}\left(\boldsymbol{\theta}|\mathcal{D}\right)}\right)d\boldsymbol{\theta}\\ &\qquad\qquad\{\text{Bayes's Theorem}\}\\ &= \mathbb{E}_{\mathsf{P}\left(\boldsymbol{\theta}|\mathcal{D}\right)}\mathbb{E}_{\mathsf{p}\left(\mathsf{Y}|\vec{\mathsf{X}},\boldsymbol{\theta}\right)}\left[\log\left(\frac{\mathsf{p}\left(\mathsf{Y}|\vec{\mathsf{X}},\boldsymbol{\theta}\right)}{\mathsf{p}\left(\mathsf{Y}|\vec{\mathsf{X}},\mathcal{D}\right)}\right)\right]\\ &\approx \frac{1}{M}\sum_{m=1}^{M}D_{\mathsf{KL}}\left[\mathsf{p}\left(\mathsf{Y}|\vec{\mathsf{X}},\boldsymbol{\theta}^{(m)}\right)\Big\|\frac{1}{M}\sum_{m=1}^{M}\mathsf{p}\left(\mathsf{Y}|\vec{\mathsf{X}},\boldsymbol{\theta}^{(m)}\right)\right].\end{aligned}$$

This shows that in the context of an ensemble, mutual information can be interpreted as the degree of variation in predictions among ensemble members. This observation aligns with the idea that high

mutual information suggests that small changes in parameters can result in significant changes in outcomes. In an ensemble setting we assume that the different members have significantly differing parameters. In the case that the mutual information between the parameters and the outcome is high, this would imply that the ensemble members will have significantly differing predictions, resulting in a high degree of disagreement. Hence this high disagreement can be seen as a indicator of high knowledge uncertainty.

By subtracting this knowledge uncertainty (mutual information) from the total uncertainty (entropy), we obtain (see Theorem 5):

$$\underbrace{\mathcal{H}\left(\mathsf{p}\left(\mathsf{Y}|\vec{\mathsf{X}},\mathcal{D}\right)\right)}_{\text{Total Uncertainty}} - \underbrace{\mathcal{I}\left(\mathsf{Y},\boldsymbol{\theta}|\vec{\mathsf{X}},\mathcal{D}\right)}_{\text{Knowledge Uncertainty}} = \underbrace{\mathbb{E}_{\mathsf{P}(\boldsymbol{\theta}|\mathcal{D})}\mathcal{H}\left(\mathsf{p}\left(\mathsf{Y}|\vec{\mathsf{X}},\boldsymbol{\theta}\right)\right)}_{\text{Data Uncertainty}}.$$

This expected remaining uncertainty in Y after taking into account the model parameters $\boldsymbol{\theta}$ encapsulates the irreducible level of uncertainty that is intrinsic to the process that generates the data. Known as *data uncertainty*, it represents the portion of uncertainty that persists despite our best efforts to model or explain it through $\boldsymbol{\theta}$.

In Figure 3.1, we explore three distinct scenarios to understand how uncertainty manifests within ensemble models.

1. **Low Uncertainty Scenario:** Figure 3.1 (A) illustrates a case where all ensemble members agree closely on their predictions, resulting in low uncertainty in the marginal distribution. In this context, both the total uncertainty, representing the overall unpredictability in the outcome, and data uncertainty, capturing the irreducible uncertainty intrinsic to the data, are low. Additionally, knowledge uncertainty is minimal, indicated by the high level of agreement among the ensemble members.

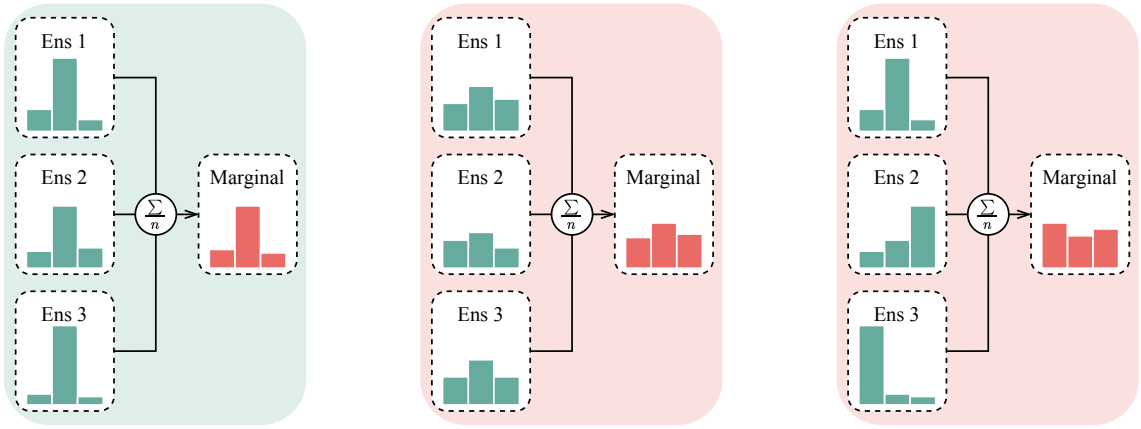
2. **High Data Uncertainty Scenario:** In Figure 3.1 (B), all ensemble members exhibit high levels of uncertainty, which is also mirrored in the marginal distribution. In this situation, both the total uncertainty and the data uncertainty are high. Interestingly, the knowledge uncertainty remains low because the ensemble members consistently produce high-uncertainty predictions, signalling a shared understanding of the underlying data's unpredictability.

3. **High Knowledge Uncertainty Scenario:** Figure 3.1 (C) depicts a more intricate case. Here, individual ensemble members show a high level of certainty in their predictions, indicating low data uncertainty. However, the marginal distribution itself displays high total uncertainty. This heightened level of total uncertainty arises primarily from knowledge uncertainty, which becomes apparent from the significant disagreement among the ensemble members. In this case, the model parameters vary enough across ensemble members to introduce a high degree of variation, or *knowledge uncertainty*, in the collective output.

While ensemble methods provide the dual advantage of calibrated predictive distributions and uncertainty estimation, their computational cost is often a limiting factor. The training of multiple models and the necessity of aggregating their predictions during inference can be both time-consuming and resource-intensive. It is particularly problematic in scenarios requiring real-time predictions or when computational resources are limited. To address these challenges without sacrificing the benefits of ensembles, various techniques have been developed to approximate the performance of an ensemble with a single, more efficient model.

3.5 Ensemble Distillation Techniques

Ensemble distillation aims to capture the essence of what an ensemble 'knows' and convey this knowledge to a solitary 'student' model. This student model is trained to mimic the ensemble's



(A) Low uncertainty. (B) High data uncertainty. (C) High knowledge uncertainty.

FIGURE 3.1: Exploring Different Sources of Uncertainty in Ensemble Models.

behaviour, effectively serving as a lightweight surrogate that approximates the ensemble's predictive distribution. The objective is to retain the calibration and uncertainty estimation capabilities of the ensemble, while significantly reducing the computational overhead. In the following sections, we delve into the methodologies and benefits of ensemble distillation, highlighting how it allows for accurate and computationally efficient model inference. We focus on two primary variants of this technique: Ensemble Distillation (EnD) and Ensemble Distribution Distillation (EnD^2).

3.5.1 Ensemble Distillation

The first approach we will explore is Ensemble Distillation (EnD), which aims to encapsulate the collective intelligence of an ensemble into a singular, more computationally efficient student model.

The foundational idea behind EnD is to distill knowledge from an ensemble of M models, $\left\{\boldsymbol{\theta}^{(1)}, \boldsymbol{\theta}^{(2)}, \cdots, \boldsymbol{\theta}^{(M)}\right\}$, into a single student model. This involves utilising a transfer dataset, denoted as $\mathcal{D}_{\text{ens}}$, containing predictions from the ensemble. Specifically, the transfer dataset consists of combinations of input features and predictive distributions, $\boldsymbol{\pi}$, from each model in the ensemble. Formally, it can be represented as:

$$\mathcal{D}_{\text{ens}} = \left\{\left(\boldsymbol{x}_i, \boldsymbol{\pi}_i^{(1)}, \boldsymbol{\pi}_i^{(2)}, \cdots, \boldsymbol{\pi}_i^{(M)}\right)\right\}_{i=1}^{N}.$$

A widely-used loss function for distillation is the Kullback-Leibler (KL) divergence, which measures how one probability distribution diverges or is different from a second, reference probability distribution. The loss function for EnD can be mathematically expressed as:

$$\mathcal{L}_{\text{EnD}}\left(\mathcal{D}_{\text{ens}}; \boldsymbol{\phi}\right) = \mathbb{E}_{\mathsf{p}\left(\vec{\mathsf{X}}|\mathcal{D}_{\text{ens}}\right)} D_{\text{KL}}\left[\mathsf{p}\left(\mathsf{Y}|\vec{\mathsf{X}}; \boldsymbol{\theta}^{(1:M)}\right) \middle\| \mathsf{p}\left(\mathsf{Y}|\vec{\mathsf{X}}; \boldsymbol{\phi}\right)\right],$$

where $\mathsf{p}\left(\mathsf{Y}|\vec{\mathsf{X}} = \boldsymbol{x}_i; \boldsymbol{\theta}^{(1:M)}\right) = \frac{1}{M}\sum_{m=1}^{M} \boldsymbol{\pi}_i^{(m)}$ denotes the marginal predictive distribution of the ensemble, and $\mathsf{p}\left(\mathsf{Y}|\vec{\mathsf{X}}; \boldsymbol{\phi}\right)$ is the predictive distribution of the student model, parameterised by $\boldsymbol{\phi}$.

Temperature Scaling

To improve the alignment of the student model's distribution with the ensemble's distribution, temperature scaling is often employed. Temperature scaling is a modification of the softmax activation function. It introducing a scaling hyper-parameter ,T, for adjusting the entropy in the resulting categorical distribution:

$$\phi_{\texttt{Softmax}}(\boldsymbol{z}; T) = \frac{\exp\left(\frac{\boldsymbol{z}}{T}\right)}{\sum_{k=1}^{K} \exp\left(\frac{z_k}{T}\right)}.$$

Here, the larger the value of T, the higher the entropy in the resulting distribution. Incorporating temperature scaling to increase entropy in both the target and predictive distributions of the student, aligns these distributions in low probability classes. This results in more effective optimisation of the KL divergence between the two distributions, thereby improving convergence (Hinton et al., 2015).

While ensemble distillation does improve efficiency, it primarily focuses on approximating the averaged predictions from the ensemble. As a result, EnD captures what could be termed as the 'mean knowledge' of the ensemble, but not the variations between ensemble members. These variations are crucial for a nuanced understanding of model uncertainty, both data and knowledge uncertainties. For a more comprehensive representation that captures both the mean and variation, we consider Ensemble Distribution Distillation ($\texttt{EnD}^2$).

3.5.2 Ensemble Distribution Distillation

Unlike traditional ensemble distillation methods where the student model is trained to mimic the mean of the ensemble's predictions, Ensemble Distribution Distillation ($\texttt{EnD}^2$) aims to take it a step further. Here, the student model is trained not just to emulate the ensemble's average predictive distribution but to model the entire distribution of predictive distributions among the ensemble members. By doing so, $\texttt{EnD}^2$ enables the distilled student model to better approximate both data and knowledge uncertainties, offering a more complete and nuanced uncertainty representation.

Let us first set the scene. The predictive distribution $\mathsf{p}\left(\mathsf{Y}|\cdot\right)$ is a categorical distribution parameterised by a vector of probabilities $\boldsymbol{\pi}$, that is:

$$\mathsf{p}\left(\mathsf{Y}|\boldsymbol{\pi}\right) \sim \texttt{Cat}\left(\boldsymbol{\pi}\right).$$

These probabilities are predicted by a model $\boldsymbol{\pi}(\boldsymbol{x}) = f(\boldsymbol{x}; \boldsymbol{\theta})$ hence we can state that they have the distribution:

$$\mathsf{P}\left(\vec{\Pi}|\vec{\mathsf{X}}; \boldsymbol{\theta}\right).$$

As a result, we can reformulate Equation 3.2 as:

$$\begin{aligned}
\mathsf{p}\left(\mathsf{Y}|\vec{\mathsf{X}}, \mathcal{D}\right) &= \int \mathsf{p}\left(\mathsf{Y}|\vec{\mathsf{X}}, \boldsymbol{\theta}\right) \mathsf{P}\left(\boldsymbol{\theta}|\mathcal{D}\right) d\boldsymbol{\theta} \\
&= \int\int \mathsf{P}\left(\mathsf{Y}, \vec{\Pi}|\vec{\mathsf{X}}, \boldsymbol{\theta}\right) d\vec{\Pi}\, \mathsf{P}\left(\boldsymbol{\theta}|\mathcal{D}\right) d\boldsymbol{\theta} \\
&= \int\int \mathsf{p}\left(\mathsf{Y}|\vec{\Pi}\right) \mathsf{P}\left(\vec{\Pi}|\vec{\mathsf{X}}, \boldsymbol{\theta}\right) d\vec{\Pi}\, \mathsf{P}\left(\boldsymbol{\theta}|\mathcal{D}\right) d\boldsymbol{\theta} \qquad \{\text{Bayes' Theorem}\} \\
&= \mathbb{E}_{\mathsf{P}(\boldsymbol{\theta}|\mathcal{D})} \mathbb{E}_{\mathsf{P}\left(\vec{\Pi}|\vec{\mathsf{X}}, \boldsymbol{\theta}\right)} \left[\mathsf{p}\left(\mathsf{Y}|\vec{\Pi}\right)\right].
\end{aligned}$$

As in Equation 3.3 we can sample from the posterior of the model parameters using an ensemble of models.

$$\begin{aligned}
&\mathsf{p}\left(\mathsf{Y}|\vec{\mathsf{X}}, \mathcal{D}\right) \approx \frac{1}{M} \sum_{m=1}^{M} \mathbb{E}_{\mathsf{P}\left(\vec{\Pi}|\vec{\mathsf{X}}, \boldsymbol{\theta}^{(m)}\right)} \left[\mathsf{p}\left(\mathsf{Y}|\vec{\Pi}\right)\right], \\
&\text{where } \boldsymbol{\theta}^{(m)} \sim \mathsf{P}\left(\boldsymbol{\theta}|\mathcal{D}\right).
\end{aligned}$$

Since each ensemble member will predict a fixed distribution $\boldsymbol{\pi}^{(m)}(\boldsymbol{x}) = f\left(\boldsymbol{x}; \boldsymbol{\theta}^{(m)}\right)$, we can state that:

$$\boldsymbol{\pi}^{(m)}(\boldsymbol{x}) \sim \mathsf{P}\left(\vec{\Pi}|\boldsymbol{x}, \mathcal{D}\right),$$

is a sample from the posterior of $\vec{\Pi}(\boldsymbol{x})$. In EnD^2, the objective of the student model is to estimate the posterior of $\vec{\Pi}(\boldsymbol{x})$. Before we continue exploring EnD^2, let us first introduce an important family of distributions known as the the Dirichlet distribution.

The Dirichlet Distribution

The Dirichlet distribution is a family of multivariate continuous probability distributions, parameterised by a vector of concentration parameters $\boldsymbol{\alpha} = \begin{bmatrix}\alpha_1 & \alpha_2 & \cdots & \alpha_K\end{bmatrix}$. Here $\boldsymbol{\alpha} \in \mathbb{R}^K$ and $\alpha_k > 0$ for all $k \in \{1, 2, \ldots, K\}$.

Consider a random vector $\vec{\mathsf{X}} = \begin{bmatrix}\mathsf{X}_1 & \mathsf{X}_2 & \cdots & \mathsf{X}_K\end{bmatrix}$, where each component $\mathsf{X}_k \in [0, 1]$ and $\sum_{k=1}^K \mathsf{X}_k = 1$. The random vector $\vec{\mathsf{X}}$ is said to follow a Dirichlet distribution with parameter vector $\boldsymbol{\alpha}$, denoted as $\vec{\mathsf{X}} \sim \mathtt{Dir}(\boldsymbol{\alpha})$, if its joint probability density function (pdf) is defined as:

$$\mathsf{P}\left(\vec{\mathsf{X}} = \boldsymbol{x}|\boldsymbol{\alpha}\right) = \frac{\Gamma(\alpha_0)}{\prod_{j=1}^K \Gamma(\alpha_j)} \prod_{k=1}^K x_k^{\alpha_k - 1}, \tag{3.4}$$

where $\alpha_0 = \sum_{k=1}^K \alpha_k$ is the sum of the concentration parameters, often referred to as the precision parameter, and $\Gamma(\cdot)$ the Gamma function.

The Dirichlet distribution exhibits the following properties, which are relevant for subsequent discussions in this chapter:

1. Expectation: The expected value of each component X_k is given by:

$$\mathbb{E}\left[\mathsf{X}_k\right] = \frac{\alpha_k}{\alpha_0}. \tag{3.5}$$

2. Log Expectation: The expected value of the natural logarithm of each component X_k is given by:

$$\mathbb{E}\left[\log\left(\mathsf{X}_k\right)\right] = \psi(\alpha_k) - \psi(\alpha_0). \tag{3.6}$$

Here $\psi(z) = \frac{d}{dz}\log\Gamma(z)$ is the digamma function.

3. Conjugacy: The Dirichlet distribution is a conjugate prior of the categorical distribution.

4. Marginal: The marginal distribution of each component X_k is a $\mathtt{Beta}(\alpha_k, \alpha_0 - \alpha_k)$ distribution. We do not delve into the details of this distribution, however it is important to note that:

$$\mathbb{E}_{\mathsf{P}(\mathsf{X}_k)}\left[\mathsf{X}_k \log \mathsf{X}_k\right] = \frac{\alpha_k}{\alpha_0}\left[\psi(\alpha_k + 1) - \psi(\alpha_0 + 1)\right] \tag{3.7}$$

Having established the foundational understanding of the Dirichlet distribution, we can now continue into its application within the framework of ensemble distribution distillation. Specifically, we configure the student model to predict a Dirichlet distribution, denoted as $\mathtt{Dir}(\boldsymbol{\alpha}(\boldsymbol{x}))$.

In this setting, $\boldsymbol{\alpha}(\boldsymbol{x})$ is defined as $e^{\boldsymbol{z}}$, where $\boldsymbol{z}$ represents the real valued output of the student model for a given input $\boldsymbol{x}$. The objective function we use is the negative log-likelihood of the transfer dataset $\mathcal{D}_{\mathsf{ens}}$, calculated with respect to the Dirichlet distribution predicted by the student model. Mathematically, this can be expressed as:

$$\mathcal{L}_{\mathsf{EnD}^2}\left(\mathcal{D}_{\mathsf{ens}}; \boldsymbol{\phi}\right) = -\mathbb{E}_{\mathsf{P}\left(\vec{\Pi}, \vec{\mathsf{X}}\middle|\mathcal{D}_{\mathsf{ens}}\right)} \log \mathsf{P}\left(\vec{\Pi}\middle|\boldsymbol{\alpha}\left(\vec{\mathsf{X}}; \boldsymbol{\phi}\right)\right).$$

Convergence Problems

Despite the theoretical appeal of using negative log-likelihood (NLL) as the optimisation objective in EnD^2, it is empirically found to result in significant convergence issues. To understand why this happens, we examine the first-order gradients, which are the first derivatives of the loss function with respect to its input variables. We examine the first order gradients using the gradient norm ratio (ρ). To better understand these convergence problems, we will consider the first-order gradients of four different loss functions: Categorical Cross-Entropy, NLL, KL Divergence, and Reverse KL Divergence.

Before diving into the other objectives, let us first establish Categorical Cross-Entropy as a baseline for efficient convergence. Although efficient, it only focuses on matching the mean of the distribution to the target, making it unsuitable for ensemble distribution distillation.

For our analysis, we define a target distribution, $\boldsymbol{\pi}^{\texttt{tgt}}$, which serves as a comparative benchmark.

$$\boldsymbol{\pi}^{\texttt{tgt}} = \begin{bmatrix} 1-\epsilon & \frac{\epsilon}{K-1} & \cdots & \frac{\epsilon}{K-1} \end{bmatrix}, \epsilon = 1e-4.$$

This results in the categorical cross entropy objective:

$$\begin{aligned}
\mathcal{L}_{\texttt{CE}}\left(\boldsymbol{x}, \boldsymbol{\pi}; \boldsymbol{\phi}\right) &= -\sum_{k=1}^{K} \pi_k^{\texttt{tgt}} \log \mathsf{p}\left(\mathsf{Y}=k|\boldsymbol{x};\boldsymbol{\phi}\right) \\
&= -\sum_{k=1}^{K} \pi_k^{\texttt{tgt}} \log \mathbb{E}_{\mathsf{P}\left(\vec{\Pi}|\boldsymbol{x};\boldsymbol{\phi}\right)}\left[\Pi_k\right] \\
&= -\sum_{k=1}^{K} \pi_k^{\texttt{tgt}} \log\left(\frac{\alpha_k}{\alpha_0}\right) \qquad \{\text{Equation 3.5}\}.
\end{aligned}$$

The first-order gradients for this loss function are then given by:

$$\frac{\partial \mathcal{L}_{\texttt{CE}}\left(\boldsymbol{x}, \boldsymbol{\pi}; \boldsymbol{\phi}\right)}{\partial z_c} = \frac{\alpha_c}{\alpha_0} - \pi_c^{\texttt{tgt}}.$$

To study the behaviour of the loss functions, we consider three commonly-encountered scenarios during the training of the student model:

1. Random Initialisation: The student model begins with random initialisation, resulting in a uniform Dirichlet distribution, $\texttt{Dir}\left(\boldsymbol{\alpha}^{\texttt{init}} = \mathbf{1}\right)$.

2. Misclassification: At this stage, the student model has not yet fully learned to correctly classify data.

$$\begin{aligned}
\boldsymbol{\alpha}^{\texttt{miss}} &= \boldsymbol{\pi}^{\texttt{miss}}\alpha_0\,, \alpha_0 = 90K \\
\boldsymbol{\pi}^{\texttt{miss}} &= \begin{bmatrix} \frac{5\epsilon}{K-1} & 1-5\epsilon & \frac{5\epsilon}{K-1} & \cdots & \frac{5\epsilon}{K-1} \end{bmatrix}.
\end{aligned}$$

3. Near-Convergence: The student model has almost converged, displaying a specific distribution of confidence among the classes.

$$\begin{aligned}
\boldsymbol{\alpha}^{\texttt{conv}} &= \boldsymbol{\pi}^{\texttt{conv}}\alpha_0\,, \alpha_0 = 90K \\
\boldsymbol{\pi}^{\texttt{conv}} &= \begin{bmatrix} 1-5\epsilon & \frac{5\epsilon}{K-1} & \frac{5\epsilon}{K-1} & \cdots & \frac{5\epsilon}{K-1} \end{bmatrix}.
\end{aligned}$$

For these scenarios we set the precision of our student model to $\alpha_0 = 90K$ for illustrative purposes, as in Ryabinin et al. (2021).

We will study the behaviour of three suitable optimisation techniques for ensemble distribution distillation. For each we will derive the objective function and its first-order gradients.

1. Negative Log Likelihood (NLL) The objective function is given by:

$$\begin{aligned}\mathcal{L}_{\text{NLL}}\left(\boldsymbol{x}, \boldsymbol{\pi}; \boldsymbol{\phi}\right) &= -\log \mathsf{P}\left(\boldsymbol{\pi} \middle| \boldsymbol{x}; \boldsymbol{\phi}\right) \\ &= -\log\left(\frac{\Gamma(\alpha_0)}{\prod_{j=1}^{K}\Gamma(\alpha_j)}\prod_{k=1}^{K}\pi_k^{\alpha_k-1}\right) \qquad \{\text{Equation 3.4}\} \\ &= \sum_{j=1}^{K}\log\Gamma(\alpha_k) - \log\Gamma(\alpha_0) - \sum_{k=1}^{K}(\alpha_k-1)\log\pi_k.\end{aligned}$$

The first-order gradients of this objective are:

$$\frac{\partial\mathcal{L}_{\text{NLL}}\left(\boldsymbol{x}, \boldsymbol{\pi}; \boldsymbol{\phi}\right)}{\partial z_c} = \left[\psi(\alpha_c) - \psi(\alpha_0) - \log\pi_c\right]\alpha_c.$$

2. KL Divergence To employ this objective function, it is essential to first have an estimate of the target posterior distribution. To tackle this issue, we make use of a proxy Dirichlet distribution denoted by $\texttt{Dir}\left(\boldsymbol{\beta}\right)$, where $\boldsymbol{\beta} = \boldsymbol{\pi}^{\text{tgt}}\beta_0$ and $\beta_0 = 100K$ (precision of $100K$ is chosen for illustrative purposes (Ryabinin et al., 2021)).

The Dirichlet distribution serves as an apt choice for this approximation for several reasons. It is designed to model the distribution of a vector of random variables that fall within the range $[0, 1]$ and sum to 1. This makes it both flexible and well-suited for representing probabilities or proportions, which aligns closely with the characteristics of a posterior distribution of the target. Once this Dirichlet-approximated posterior is established, the KL divergence objective can then be derived as:

$$\begin{aligned}\mathcal{L}_{\text{KL}}\left(\boldsymbol{x}, \boldsymbol{\beta}; \boldsymbol{\phi}\right) &= D_{\text{KL}}\left[\mathsf{P}\left(\vec{\Pi}\middle|\boldsymbol{\beta}\right)\middle\|\mathsf{P}\left(\vec{\Pi}\middle|\boldsymbol{\alpha}(\boldsymbol{x})\right)\right], \text{where } \mathsf{P}\left(\vec{\Pi}\middle|\boldsymbol{\beta}\right) \sim \texttt{Dir}\left(\boldsymbol{\beta}\right), \mathsf{P}\left(\vec{\Pi}\middle|\boldsymbol{\alpha}(\boldsymbol{x})\right) \sim \texttt{Dir}\left(\boldsymbol{\alpha}(\boldsymbol{x})\right) \\ &= \mathbb{E}_{\mathsf{p}\left(\vec{\Pi}\middle|\boldsymbol{\beta}\right)}\left[\log\mathsf{P}\left(\vec{\Pi}\middle|\boldsymbol{\beta}\right)\right] - \mathbb{E}_{\mathsf{P}\left(\vec{\Pi}\middle|\boldsymbol{\beta}\right)}\left[\log\mathsf{P}\left(\vec{\Pi}\middle|\boldsymbol{\alpha}(\boldsymbol{x})\right)\right] \\ &= \log\Gamma(\beta_0) - \sum_{k=1}^{K}\log\Gamma(\beta_k) - \log\Gamma(\alpha_0) + \sum_{k=1}^{K}\log\Gamma(\alpha_k) + \sum_{k=1}^{K}(\beta_k-\alpha_k)\mathbb{E}_{\mathsf{P}\left(\vec{\Pi}\middle|\boldsymbol{\beta}\right)}\left[\log\left(\vec{\Pi}\right)\right] \\ &= \log\Gamma(\beta_0) - \sum_{k=1}^{K}\log\Gamma(\beta_k) - \log\Gamma(\alpha_0) + \sum_{k=1}^{K}\log\Gamma(\alpha_k) + \sum_{k=1}^{K}(\beta_k-\alpha_k)\left(\psi(\beta_k) - \psi(\beta_0)\right) \\ &\qquad\qquad \{\text{Equation 3.6}\}.\end{aligned}$$

The first-order gradients of this objective is:

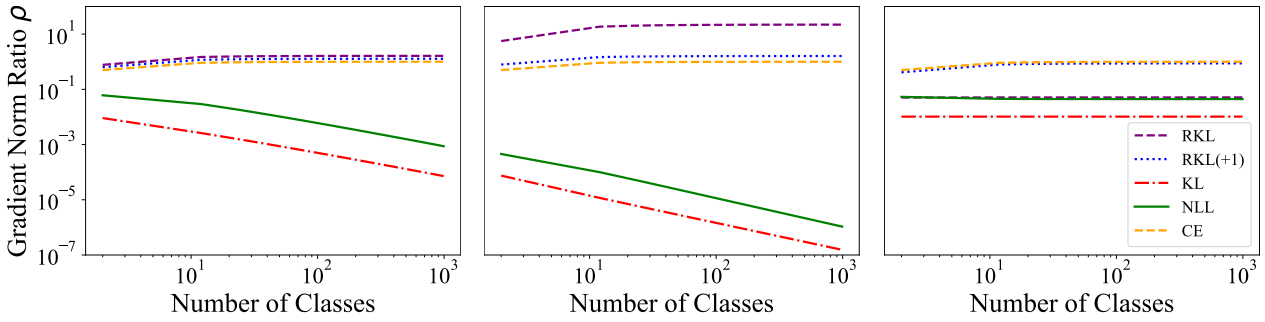
$$\frac{\partial\mathcal{L}_{\text{KL}}\left(\boldsymbol{x}, \boldsymbol{\beta}; \boldsymbol{\phi}\right)}{\partial z_c} = \left[\psi(\alpha_c) - \psi(\alpha_0) - \psi(\beta_c) + \psi(\beta_0)\right]\alpha_c.$$

3. Reverse KL Divergence Similarly to forward KL divergence, the objective function is defined as:

$$\begin{aligned}\mathcal{L}_{\text{RKL}}\left(\boldsymbol{x}, \boldsymbol{\beta}; \boldsymbol{\phi}\right) &= D_{\text{KL}}\left[\mathsf{P}\left(\vec{\Pi}\middle|\boldsymbol{\alpha}(\boldsymbol{x})\right)\middle\|\mathsf{P}\left(\vec{\Pi}\middle|\boldsymbol{\beta}\right)\right] \\ &= \log\Gamma(\alpha_0) - \sum_{k=1}^{K}\log\Gamma(\alpha_k) - \log\Gamma(\beta_0) + \sum_{k=1}^{K}\log\Gamma(\beta_k) + \sum_{k=1}^{K}(\alpha_k-\beta_k)\left(\psi(\alpha_k) - \psi(\alpha_0)\right).\end{aligned}$$

The first-order gradients of this objective is:

$$\frac{\partial\mathcal{L}_{\text{RKL}}\left(\boldsymbol{x}, \boldsymbol{\beta}; \boldsymbol{\phi}\right)}{\partial z_c} = \left[(\alpha_c-\beta_c)\psi'(\alpha_c) - (\alpha_0-\beta_0)\psi'(\alpha_0)\right]\alpha_c.$$



(A) Random Initialisation. (B) Misclassification. (C) Near-Convergence.

FIGURE 3.2: Comparison of Gradient Norm Ratios Across Loss Functions for Varying Numbers of Classes in Three Distinct Scenarios: Random Initialisation, Mid-Training Misclassification, and Near-Convergence. The loss functions analysed include Categorical Cross-Entropy (CE), Dirichlet Negative Log Likelihood (NLL), and both Forward and Reverse Kullback-Leibler Divergence (KL and RKL).

Here, $\psi'(\cdot)$ is the derivative of the digamma function.

Directly analysing the first-order gradients may not provide a complete picture of why NLL suffers from convergence issues. Therefore, we study the gradient norm ratio ρ, to better compare the behaviour of these different loss functions.

$$\rho = \frac{K-1}{K}\left|\frac{\partial \mathcal{L}}{\partial z_1}\right| \Big/ \sum_{k=2}^{K}\left|\frac{\partial \mathcal{L}}{\partial z_k}\right|.$$

A ρ value close to 1 that remains constant or increases with a rise in K suggests that the loss function effectively guides the model to focus on high-probability classes without getting distracted by the long tail of of the distribution.

Based on the observations in Figure 3.2, we note that the gradient norm ratio for NLL is significantly less than one and decreases as the number of classes increases. This suggests that NLL focuses first on fitting the tails of the distribution before optimising the target class, resulting in slow convergence. The KL objective suffers from the same problem as NLL. In contrast, Reverse KL divergence behaves more like the categorical cross-entropy objective, the gradient norm ratio is close to one and increases as the number of classes increases. This suggests that they focus on the target class rather than on the tails of the distribution, and as a result aids faster convergence. We also observe that the reverse KL objective $D_{\mathrm{KL}}\left[\mathtt{Dir}\left(\boldsymbol{\alpha}+\mathbf{1}\right)\middle\|\mathtt{Dir}\left(\boldsymbol{\beta}+\mathbf{1}\right)\right]$ most closely resembles the behaviour of categorical cross entropy loss (ibid.).

In summary, while Negative Log Likelihood and Forward KL Divergence may pose challenges for efficient convergence, Reverse KL Divergence shows promise for fast and effective training. Its behaviour closely resembles that of the well-understood Categorical Cross-Entropy loss, making it a promising candidate for further investigation. Having discussed the gradient behaviours of these loss functions, we now turn our attention to the specifics of implementing Reverse KL Divergence through the use of a proxy Dirichlet distribution as proposed by Ryabinin et al. (ibid.).

The Proxy Dirichlet Distribution

In the context of the Reverse KL objective, a proxy Dirichlet distribution is required for representing the ensemble of models. This particular distribution is denoted as $\mathtt{P}\left(\vec{\Pi}|\boldsymbol{\beta}+\mathbf{1}\right) \sim \mathtt{Dir}\left(\boldsymbol{\beta}+\mathbf{1}\right)$, where $\boldsymbol{\beta} = \hat{\boldsymbol{\pi}}(\boldsymbol{x}) \cdot \beta_0$. Here, the ensemble's marginal predictive distribution can be used to estimate the mean $\hat{\boldsymbol{\pi}}(\boldsymbol{x})$.

Using this estimated mean, we can approximate the precision parameter β_0 via maximum likelihood estimation, employing Stirling's approximation (see Appendix Theorem 5). This is mathematically expressed as:

$$\hat{\beta}_0(\boldsymbol{x}) = \frac{K-1}{2\sum_{k=1}^{K} \hat{\pi}_k(\boldsymbol{x}) \left(\log \hat{\pi}_k(\boldsymbol{x}) - \log \pi_k(\boldsymbol{x})\right)}$$

For the unknown $\log \pi_k(\boldsymbol{x})$, we leverage the ensemble to estimate it as $\frac{1}{M}\sum_{m=1}^{M} \log \hat{\pi}_k^{(m)}(\boldsymbol{x})$. The precision parameter serves to capture the variation within the ensemble, which supplements standard ensemble distillation techniques.

To maintain numerical stability during training, both the proxy target and the model-predicted parameters are constrained to be greater than 1. This helps prevent the distribution from becoming sparse.

Given the formulation of the EnD^2 objective, understanding how to estimate the model's uncertainty is the next step. In the next section we delve into how uncertainties are estimated within this framework, leveraging the properties of the Dirichlet distribution.

Uncertainty Estimation Using the EnD^2 Student

The exploration of uncertainties is a logical next step, given our discussion on the EnD^2 loss components. To fully utilise EnD^2, it is vital to understand three key types of uncertainties: the total uncertainty, knowledge uncertainty, and data uncertainty.

In this context, the model's predictions are represented by a Dirichlet distribution, denoted as $\mathsf{P}\left(\vec{\Pi}\middle|\vec{\mathsf{X}}, \boldsymbol{\phi}\right) \sim \mathtt{Dir}(\boldsymbol{\alpha})$. This distribution serves as the backbone for estimating the posterior $\mathsf{P}\left(\vec{\Pi}\middle|\vec{\mathsf{X}}, \mathcal{D}\right)$. Given this estimation we derive the total uncertainty estimate of the model as:

$$\begin{aligned}
\mathcal{H}\left(\mathsf{p}\left(\widehat{\mathsf{Y}\middle|\vec{\mathsf{X}}, \mathcal{D}}\right)\right) &= \mathcal{H}\left(\mathbb{E}_{\mathsf{P}\left(\vec{\Pi}\middle|\vec{\mathsf{X}}, \boldsymbol{\phi}\right)}\left[\mathsf{p}\left(\mathsf{Y}\middle|\vec{\Pi}\right)\right]\right) \\
&= -\sum_{k=1}^{K} \mathbb{E}_{\mathsf{P}\left(\vec{\Pi}\middle|\vec{\mathsf{X}}, \boldsymbol{\phi}\right)}\left[\Pi_k\right] \log\left(\mathbb{E}_{\mathsf{P}\left(\vec{\Pi}\middle|\vec{\mathsf{X}}, \boldsymbol{\phi}\right)}\left[\Pi_k\right]\right) \\
&= -\sum_{k=1}^{K} \frac{\alpha_k}{\alpha_0} \log\left(\frac{\alpha_k}{\alpha_0}\right) \qquad \{\text{Equation 3.5}\}.
\end{aligned}$$

Similarly we can also derive the knowledge uncertainty estimate as:

$$\begin{aligned}
\mathcal{I}\left(\widehat{\mathsf{Y}, \boldsymbol{\phi}\middle|\vec{\mathsf{X}}, \mathcal{D}}\right) &= \mathbb{E}_{\mathsf{P}\left(\vec{\Pi}\middle|\vec{\mathsf{X}}, \boldsymbol{\phi}\right)} \mathbb{E}_{\mathsf{p}\left(\mathsf{Y}\middle|\vec{\Pi}\right)}\left[\log\left(\frac{\mathsf{p}\left(\mathsf{Y}\middle|\vec{\Pi}\right)}{\mathsf{p}\left(\mathsf{Y}\middle|\vec{\mathsf{X}}, \mathcal{D}\right)}\right)\right] \\
&= \sum_{k=1}^{K} \mathbb{E}_{\mathsf{P}\left(\vec{\Pi}\middle|\vec{\mathsf{X}}, \boldsymbol{\phi}\right)}\left[\Pi_k \log \Pi_k\right] - \sum_{k=1}^{K} \mathbb{E}_{\mathsf{P}\left(\vec{\Pi}\middle|\vec{\mathsf{X}}, \boldsymbol{\phi}\right)}\left[\Pi_k\right] \log\left(\frac{\alpha_k}{\alpha_0}\right) \\
&= \sum_{k=1}^{K} \frac{\alpha_k}{\alpha_0}\left[\psi(\alpha_k + 1) - \psi(\alpha_0 + 1)\right] - \sum_{k=1}^{K} \frac{\alpha_k}{\alpha_0} \log\left(\frac{\alpha_k}{\alpha_0}\right) \qquad \{\text{Equation 3.7}\} \\
&= -\sum_{k=1}^{K} \frac{\alpha_k}{\alpha_0} \log\left(\frac{\alpha_k}{\alpha_0}\right) + \sum_{k=1}^{K} \frac{\alpha_k}{\alpha_0}\left[\psi(\alpha_k + 1) - \psi(\alpha_0 + 1)\right] \\
&= \mathcal{H}\left(\mathsf{p}\left(\widehat{\mathsf{Y}\middle|\vec{\mathsf{X}}, \mathcal{D}}\right)\right) + \sum_{k=1}^{K} \frac{\alpha_k}{\alpha_0}\left[\psi(\alpha_k + 1) - \psi(\alpha_0 + 1)\right].
\end{aligned}$$

Lastly, we derive the data uncertainty estimate as:

$$\begin{aligned}\mathbb{E}_{\mathsf{P}(\boldsymbol{\theta}|\mathcal{D})}\widehat{\mathcal{H}\left(\mathsf{p}\left(\mathsf{Y}|\vec{\mathsf{X}},\boldsymbol{\theta}\right)\right)} &= \mathcal{H}\left(\mathsf{p}\left(\widehat{\mathsf{Y}|\vec{\mathsf{X}},\mathcal{D}}\right)\right) - \mathcal{I}\left(\widehat{\mathsf{Y},\boldsymbol{\phi}|\vec{\mathsf{X}},\mathcal{D}}\right) \\ &= -\sum_{k=1}^{K}\frac{\alpha_k}{\alpha_0}\left[\psi(\alpha_k+1)-\psi(\alpha_0+1)\right].\end{aligned}$$

Here we have meticulously unpacked the formulas for estimating total, knowledge, and data uncertainties when employing the EnD^2 student model. Each of these components of uncertainty provides a deeper understanding of the models limitations and capabilities.

In conclusion, the Ensemble Distribution Distillation (EnD^2) framework, anchored by its Reverse KL objective and the innovative use of a proxy Dirichlet distribution, has shown itself to be a robust method for model training. It permits a seamless integration of uncertainty estimations, thereby enhancing the model's interpretability and reliability. The treatment of uncertainties, total, knowledge, and data, yields a multi-faceted view of the model's predictions, enriching our capacity for decision-making and risk assessment. In summary, the incorporation of these elements in the EnD^2 framework sets the stage for further exploration and implementation.

3.6 Uncertainty-based Active Learning

Active learning is a machine learning framework that identifies under-represented scenarios in labelled data and interactively queries an annotator (Cohn et al., 1996). This framework employs an acquisition function to select the most beneficial data points for querying. This function estimates the potential improvement in performance resulting from an observed label. These acquisition functions typically depend on prediction uncertainty (Houlsby et al., 2011), data space coverage (Sener and Savarese, 2018), variance reduction (Johansson et al., 2007), or topic popularity (Iovine et al., 2022).

Several works, including Freeman (1965), Gal et al. (2017), Houlsby et al. (2011), and Malinin (2019) highlight the importance of epistemic uncertainty in active learning. This is intuitive because epistemic uncertainty essentially signals what the model does not know but could potentially learn from new data. To focus on the most informative data points, uncertainty-based active learning approaches often rely on uncertainty-based acquisition functions. These functions evaluate the model's current understanding (or lack thereof) for each unlabelled data point and decide which ones should be queried next. We discuss some of the widely-used acquisition functions below.

3.6.1 Acquisition Functions

Consider a classification problem with input features $\vec{\mathsf{X}}$ and possible outcomes $\mathsf{Y} \in \{1, 2, \cdots, K\}$. To solve this problem we train a classification model using a set of labelled examples, called the training dataset $\mathcal{D}_{\texttt{labelled}} = \{(\boldsymbol{x}_1, \boldsymbol{y}_1), (\boldsymbol{x}_2, \boldsymbol{y}_2), ..., (\boldsymbol{x}_N, \boldsymbol{y}_N)\}$. The acquisition function for a data point with input features $\boldsymbol{x}$ is denoted by $a(\boldsymbol{x})$. In this section we introduce some popular uncertainty based acquisition functions.

Least confidence The least confidence strategy selects data points where the model is least confident in its most likely class prediction. For a given data point $\boldsymbol{x}$:

$$a(\boldsymbol{x}) = 1 - \underbrace{\max_c \mathsf{p}\left(\mathsf{Y} = c|\boldsymbol{x}, \mathcal{D}_{\texttt{labelled}}\right)}_{\text{Confidence score}}.$$

The instances with the highest acquisition values are considered as candidates for labelling.

Total uncertainty This strategy uses the total uncertainty in the model's prediction as the acquisition function. The entropy of the predictive distribution can be used to measure the total uncertainty (Shannon, 1948). For a given data point $\boldsymbol{x}$:

$$a(\boldsymbol{x}) = \mathcal{H}\left(\mathsf{p}\left(\mathsf{Y}\middle|\boldsymbol{x}, \mathcal{D}_{\texttt{labelled}}\right)\right).$$

In other words, we query points where to total uncertainty is highest.

Bayesian active learning by disagreement (BALD) aims to select instances with the highest knowledge uncertainty. This is measured by the mutual information between the predictive distribution and the posterior distribution of the model parameters. In simpler terms, BALD seeks instances that are not only uncertain in prediction but also highly informative about the model parameters (Houlsby et al., 2011). Mathematically, the acquisition function is expressed as:

$$a(\boldsymbol{x}) = \mathcal{I}\left(\mathsf{Y}, \boldsymbol{\theta}\middle|\vec{\mathsf{X}}, \mathcal{D}_{\texttt{labelled}}\right).$$

In summary, active learning leverages model uncertainty to make intelligent queries, aiming to accelerate the learning process and reduce annotation costs. Different acquisition functions serve as heuristics for selecting the most informative instances, and their effectiveness can vary depending on the application and nature of the data. Understanding and utilising these uncertainty measures is crucial for designing effective active learning systems.

3.7 Conclusion

The chapter laid the groundwork for understanding uncertainty estimation in deep learning by delineating between data-driven (aleatoric) and knowledge-based (epistemic) uncertainties. We discussed calibration methods like modified objective functions and ensemble techniques to improve the calibration of uncertainty estimates. The chapter also introduced ensemble distillation as a means to achieve effective uncertainty estimation while reducing computational overhead. Finally, we examined the role of uncertainty in active learning, particularly how it can be harnessed through uncertainty-based acquisition functions to select the most informative data points for labelling. These foundational concepts and techniques set the stage for deeper exploration into the impact of uncertainty on downstream tasks and active learning, to be covered in Chapters 5-7.

Chapter 4

Dialogue State Tracking

4.1 Overview

In a typical modular task-oriented dialogue system, the role of the dialogue state/belief tracker is to keep track of the dialogue state. This state consists of the user's goal, the user's latest actions as well as other important features such as completed bookings and database query results. Young et al. (2007) defined the Hidden Information State (HIS) as the set of minimum requirements for the dialogue state. In Figure 4.1, we detail the key components of a dialogue belief state:

User Goal: It features distributions that represent the user's intent and the degree of uncertainty linked with that intent. This goal evolves with the dialogue, reflecting updates at each conversational turn and thus serves as a representation of the dialogue context. It is the belief tracker's task to update this based on user inputs and system actions.

Latest User and System Actions: These record the most recent interactions and provide essential context, offering insights into the ongoing flow of the conversation.

Database Queries: The belief tracker, utilising the user's specified goals or constraints, queries the database. The retrieved database results contains the number of entities that match with the user's requirements.

Booking Information: This section logs all the reservations or bookings initiated by the system during the conversation. It is a dynamic component, updated by the dialogue belief tracker in response to system actions, and forms an important part of the dialogue context.

The belief tracker's primary role, as evident, is to monitor, update, and maintain the dialogue belief state. The system's subsequent response is entirely based on this state, highlighting the importance of the dialogue state tracker's accuracy for the overall success of the system. In this section, we delve into the evolution of tracking methodologies, starting with generative approaches.

4.1.1 Generative Approaches to Dialogue Modelling

Early dialogue system models portrayed the interaction using the framework of a Markov Decision Process (MDP) (Levin et al., 1998). In this framework, the state-space is defined by all *dialogue states*, which typically encompass the user's requests and constraints, outlining the user's intentions or goals. The policy aims to identify an optimal strategy for the MDP based on the inferred dialogue state. This model assumes that the updated state $\boldsymbol{s}_{t+1}$ depends solely on the current state $\boldsymbol{s}_t$ and the system action a_t, leading to the transition probabilities $\mathsf{p}\left(\boldsymbol{s}_{t+1}|\boldsymbol{s}_t, a_t\right)$.

The limitation of the MDP model is that the dialogue state is not directly observable, which precludes the use of a fully observable MDP. Statistical dialogue models overcome this limitation by introducing a belief state, a distribution over all possible dialogue states (Williams and Young, 2007). A statistical generalisation of the MDP is the partially observable Markov decision process (POMDP), depicted in Figure 4.2, which models the process of generating the observed user action o_{t+1} based on

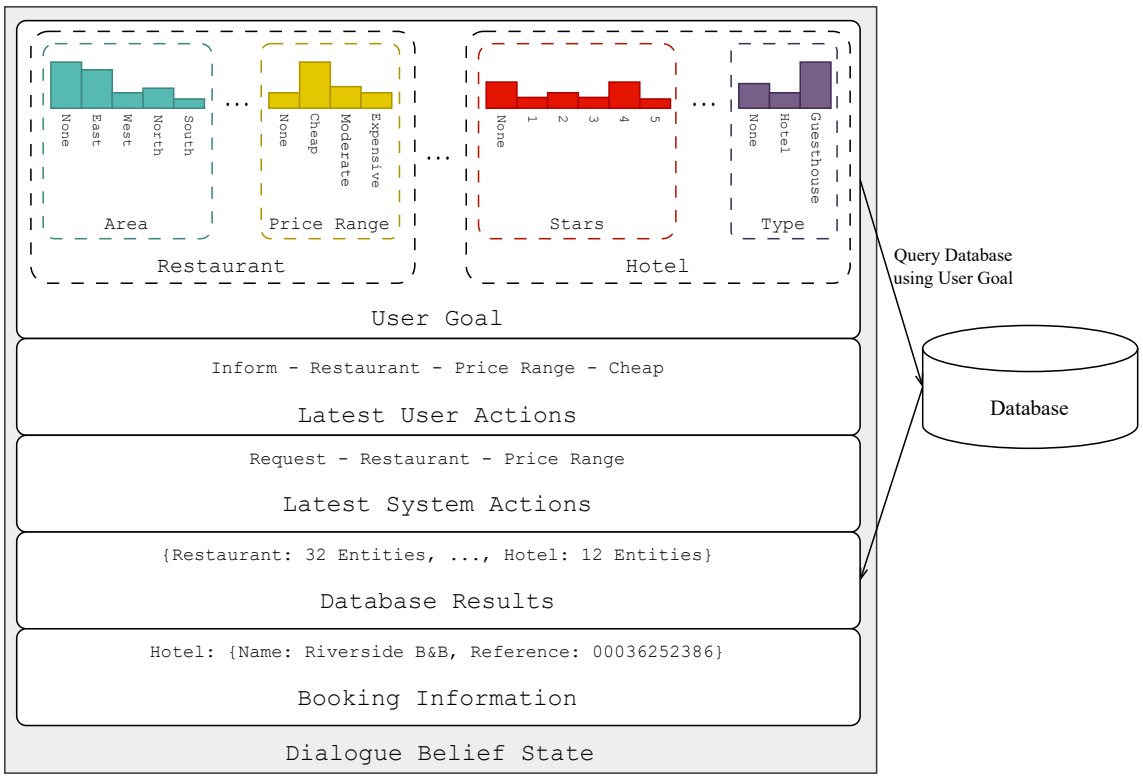


FIGURE 4.1: Components of the Dialogue Belief State in a Task-Oriented Dialogue System. The User Goal captures the desired criteria set by the user, such as location and price preferences. The Latest User Actions and System Actions record the recent turn-by-turn history of the conversation, providing essential context. The system queries the Database using the User Goal to retrieve relevant information. The returned Database Results contain matching entities. The Booking Information showcases specific reservations or bookings made during the dialogue. The Dialogue Belief State consolidates all this information, summarising the system's current understanding of the conversation.

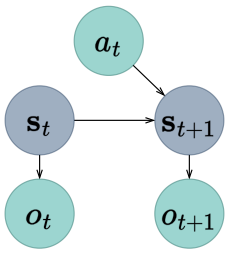


FIGURE 4.2: Graphical model of the conditional dependencies in a task-oriented dialogue system. The variables s_t, a_t, and o_t represent the unobserved state of the dialogue (in grey), the observed system and user actions (in green) at time step t.

the unobserved state s_{t+1}. The POMDP model generates a distribution over possible realisations of the unobserved state, known as the belief state, defined as:

$$\begin{aligned} \mathsf{b}(s_{t+1}) &= \sum_{s_t} \mathsf{p}\left(s_{t+1}|o_{t+1}, s_t, a_t\right) \mathsf{b}(s_t) \\ &\propto \sum_{s_t} \mathsf{p}\left(s_{t+1}, o_{t+1}|s_t, a_t\right) \mathsf{b}(s_t) \qquad \{\text{Bayes Theorem}\} \\ &= \mathsf{p}\left(o_{t+1}|s_{t+1}\right) \sum_{s_t} \mathsf{p}\left(s_{t+1}|s_t, a_t\right) \mathsf{b}(s_t), \end{aligned}$$

The key problem with the POMDP model is that we need to sum over all possible states to update

our belief state after each user utterance, which could be an incredibly large number of states. The "problem of summation over the unobservable state" in a POMDP dialogue belief tracker refers to this computational complexity of keeping track of such a large number of states. Even slight uncertainty or ambiguity in user utterances can lead to substantial state space expansion. Considering all possible interpretations and maintaining a belief state over them quickly becomes intractable as the dialogue progresses. An alternative to generative models are discriminative models, which are significantly less computationally complex as they do not need to sum over a large state space (Henderson et al., 2014a, 2013).

4.1.2 Discriminative Approaches to Tracking

Discriminative approaches aim to model the conditional distribution of the state, $\mathrm{p}\left(\boldsymbol{s}_{t+1}|o_{t+1}, \boldsymbol{s}_t, a_t\right)$, directly. Unlike generative models, these methods often model the marginal distributions of state subsets independently. For instance, in dialogue state/belief tracking, each `domain-slot` pair representing user preferences in the user goal is modelled independently, as illustrated in Figure 4.1. Discriminative approaches do not model the underlying process of how states produce observations, they simply learn a mapping from the input (an observation) to output (state). Therefore, these methods utilise features associated with the user action, system action, and the previous state. Initial discriminative approaches to state tracking used features such as word counts or confidence scores of the language understanding model (Metallinou et al., 2013). Based on these features, models like ranking algorithms or decision trees were used to predict the belief state probabilities (Metallinou et al., 2013; Williams, 2014).

Deep neural networks for dialogue state tracking were first introduced by Henderson et al. (2013). Their method employed a combination of features from the language understanding module, the user actions, system action, and the state. As anticipated, deep learning approaches outperformed decision trees and ranking models. These techniques were further improved by incorporating recurrent neural networks (RNNs), which enabled the model to track information more effectively. The RNN-based belief tracker proposed by Henderson et al. (2014b) demonstrated that using the current turn information together with RNN's internal memory could compete with approaches that combined features from the entire dialogue history.

Neural network-based approaches hold a significant computational advantage over POMDP-based methods, which makes them more suitable for complex settings with large ontologies. Additionally, neural network models have the ability to infer the state directly from word-level observations. This functionality can potentially eliminate the need for a separate Natural Language Understanding (NLU) module, further simplifying the dialogue system's architecture. Consequently, these strengths make neural network approaches particularly valuable for deploying efficient dialogue state trackers.

4.2 Integrated Approaches to Tracking

Integrated approaches unify the tasks of natural language understanding and state tracking into one model, extracting the dialogue state directly from the word-level features (Henderson et al., 2014b; Mrkšić et al., 2017). Early approaches consisted of a language understanding component, which extracted the information mentioned in the current turn of the dialogue, and a belief state update component, which combines current turn and dialogue context information. Whilst these components could be jointly trained, they faced an information bottleneck. This bottleneck is particularly apparent when the prediction of one aspect relied on information from a previous turn, which may not have been adequately preserved in the belief state (Ren et al., 2018). To overcome this, the language understanding component can be conditioned on the current turn as well as the full dialogue context (Heck et al., 2020b; Kim et al., 2020; Zhang et al., 2020). This is, however, an inefficient solution, especially for lengthy dialogues, as this requires the model to re-process large amounts of information at each turn. To overcome this, information tracking can be performed on a latent level (tracking an unobserved

internal representation of the information present in a dialogue turn). Using the context conditioned *latent state*, unobserved internal representation of the state of the dialogue, the belief state distribution is predicted (Lee et al., 2019; Ren et al., 2018; Shan et al., 2020). Such latent states allows the model to be more adaptable and learn complex patterns from the dataset. It facilitates the automatic extraction of essential features enhancing the model's performance.

4.2.1 The Role of Word Embeddings

Models which infer the dialogue state directly from word-level utterances relies on word vector representations which capture the semantic meanings of words. The idea of capturing semantics through learned representation spaces has been a well-researched area since it was first introduced by Bengio et al. (2003) and Schütze (1998). The field has transitioned from count-based methods like Bag of Words (BoW) (Luhn, 1957) and Term Frequency-Inverse Document Frequency (TF-IDF) (Sparck Jones, 1972) to more advanced learned vector representation techniques. This shift was initiated by early word representation models such as Word2Vec (Mikolov et al., 2013) and GloVe (Pennington et al., 2014). DST models then advanced to incorporate complex, contextualised embeddings from models such as ELMo (Peters et al., 2018), Transformer LM (Vaswani et al., 2017), BERT (Devlin et al., 2019), T5 (Raffel et al., 2020), and GPT (Radford et al., 2018, 2019). These vector representation-based methods are commonly referred to as language models, due to their ability to model the nuances of language. The latter of these models are often referred to as large language models due to the large number of parameters, large training data sizes and computational complexity.

4.2.2 Recent Advances

Recent advancements in Dialogue State Tracking (DST) largely hinge on the use of large language models (LLMs). These LLM-based DST models can be broadly classified into three distinct groups: generation models, span-prediction models and picklist style classification models.

Generation models frame the DST task as a natural language generation problem. The goal of generation-based DST models is to generate the JSON string representation of the dialogue state, `{domain:{slot:value}}`, using the full dialogue history as context. Such generation based DST models predict the complete state at each turn rather than tracking the dialogue state (Lin et al., 2021a; Lin et al., 2021b).

One of the main disadvantages of these approaches revolves around the model's potential to hallucinate information. Such models are not bound by the constraints of the model ontology or dialogue utterances. They could incorporate any generated information into the state, even when this information is never mentioned by the user, a phenomenon known as 'hallucination'. This unrestrained flexibility can lead to models inventing or misrepresenting information.

Additionally, auto-regressive models, which generate sequences token by token based on previously generated tokens, introduce a degree of complexity when extracting a belief state. These models are required to generate characters to yield a valid JSON string output, creating a situation where the generation probabilities are conditioned on tokens that may not hold relevance for the state itself. This inherently makes the process of estimating uncertainty challenging for two main reasons:

Dependence on Irrelevant Tokens: When generating a JSON string, the model is required to generate tokens like punctuation and syntax-specific characters (e.g., `"{"`, `"}"`, `":"`, etc.) that are necessary for the JSON format but do not contribute to the semantic meaning of the dialogue state. Thus, the prediction of subsequent tokens (which could be essential for the dialogue state) becomes conditioned on these irrelevant tokens. This could potentially introduce noise or distort the probabilities assigned to important tokens, making the model's uncertainty estimates less reliable.

Order Dependency: The probability of a given value in the generated state is not only dependent on the semantics of the dialogue but also on the way the model structures the state. In other words, the order in which the model decides to generate the components of the state could affect the probabilities

associated with each part. This adds another layer of complexity to uncertainty estimation as the same dialogue state can be represented in multiple ways, depending on the order of the elements.

As a result, deriving a belief state distribution and relevant uncertainty estimates from such models is an area of future research.

Span-prediction models aim to tackle the value-dependence issue of DST models. Instead of generating the state in natural language form, these models aim to identify the `value` within a user's utterance (Heck et al., 2020b; Kim et al., 2020). This approach, exemplified by the TripPy model (Heck et al., 2020b), has proven to be highly effective, earning the model state-of-the-art status in the DST task.

Despite their value independence, these models do have additional requirements when it comes to data labeling. Rather than relying on traditional dialogue state labels found in datasets like MultiWOZ (Budzianowski et al., 2018), they necessitate labels that indicate the start and end positions of `values` within the dialogue. This need arises because these models predict the span of words representing the `value`.

To partially alleviate this demand for span-labels, Heck et al. (2022) introduced a classification head based on the weighted sum of representations, the weights being determined by the span prediction scores. This adaptation allows for fine-tuning of the model on data devoid of span-annotations.

However, this solution is not without its limitations. The classification head is predominantly used for training, while the value-independent span-prediction mechanism is primarily used for inference. Consequently, there is a unsolved challenge in deriving a belief state distribution and relevant uncertainty estimates from span-prediction models.

Picklist models are fundamentally discriminative classification neural networks. They function by choosing a `value` for each `domain-slot` pairing from a pre-existing set of candidates (set of values defined in the ontology). The prediction is determined by a similarity score that compares the model's internal representation with that of a `value` candidate (Lee et al., 2019; Zhang et al., 2020). Consequently, these models naturally generate belief state distributions, $\mathsf{p}\left(\boldsymbol{s}_{t+1} | o_{t+1}, \boldsymbol{s}_t, a_t\right)$.

However, these models are not without their limitations. They are prone to over fitting to training-set candidates. Over fitting results in subpar performance, when exposed to candidates not encountered during training. This issue frequently arises in real-world applications where the pool of potential values often includes many unseen values during the training phase. The accuracy of the similarity metric used and the representations of possible `value` candidates also significantly influence performance.

Due to their tendency to over fit to the training data, these models often display an overconfidence, where the predicted probabilities do not reflect the accuracy of their predictions (Gal and Ghahramani, 2016; Guo et al., 2017). The problem of estimating accurate uncertainty estimates using such models is an open research problem.

The dialogue belief tracking models used in the remainder of this work is based on a picklist style model, SUMBT (Lee et al., 2019). We chose this model because it serves as an excellent baseline for dialogue belief tracking, inherently capable of generating belief state distributions. Our goal is to enhance its calibration, enabling a more refined exploration of calibration techniques and the significance of uncertainty. This models inherent capabilities allow us to focus on these aspects without the constraints and complexities introduced by span-prediction or generation-based models. In the following section we will introduce the mechanics of this model in detail to set the scene for the works to follow.

4.3 The Slot Utterance Matching Approach to Belief Tracking (SUMBT)

The SUMBT model is composed of three critical components: slot utterance matching, context tracking, and user goal prediction. This section aims to provide a detailed description of each component while highlighting opportunities for potential improvements. Before introducing the SUMBT model for dialogue belief tracking, we will formalise the task of dialogue belief tracking.

4.3.1 The Dialogue Belief Tracking Task

To formalise the dialogue belief tracking task, we define the ontology of a task-oriented dialogue system (Section 1.1.1). The dialogue ontology $\mathcal{O}$ consists of a set of M `domain-slot` pairs $\{\mathtt{s}_1, \mathtt{s}_2, \cdots, \mathtt{s}_M\}$ and a set of plausible values $\mathcal{V}_{\mathtt{s}_m}$ for each $\mathtt{s}_m$. The goal of the dialogue belief tracker (DBT) is to infer the user's intention for each $\mathtt{s}_m$ by predicting a probability distribution over the plausible values. Notably, each set of plausible values, $\mathcal{V}_{\mathtt{s}_m}$, includes the `not_mentioned` value, indicating that a specific `domain-slot` pair is not part of the user's goal. This allows for computing the model's confidence for slots not present in the goal.

We also formalise the dialogue state, which contains the `value` for each `domain-slot`, in every dialogue turn. The dialogue state at turn t in dialogue i is represented as:

$$\mathcal{B}_{t,i} = \left\{ \left(\mathtt{s}_m, v_{t,i}^{\mathtt{s}_m} \right) \right\}_{\mathtt{s}_m \in \mathcal{O}} ,$$

where $v_{t,i}^{\mathtt{s}_m}$ denotes the `value` for the `domain-slot` pair $\mathtt{s}_m$ at turn t in dialogue i. A resulting dataset for dialogue belief tracking can be expressed as:

$$\mathcal{D} = \left\{ \left(\boldsymbol{u}_{t,i}^{\mathtt{usr}}, \boldsymbol{u}_{t-1,i}^{\mathtt{sys}}, \mathcal{B}_{t,i} \right)_{t=1}^{T_i} \right\}_{i=1}^{D} .$$

Each dialogue consists of T_i turns, where user and system utterances at turn t in dialogue i are denoted as $\boldsymbol{u}_{t,i}^{\mathtt{usr}}$ and $\boldsymbol{u}_{t,i}^{\mathtt{sys}}$, respectively.

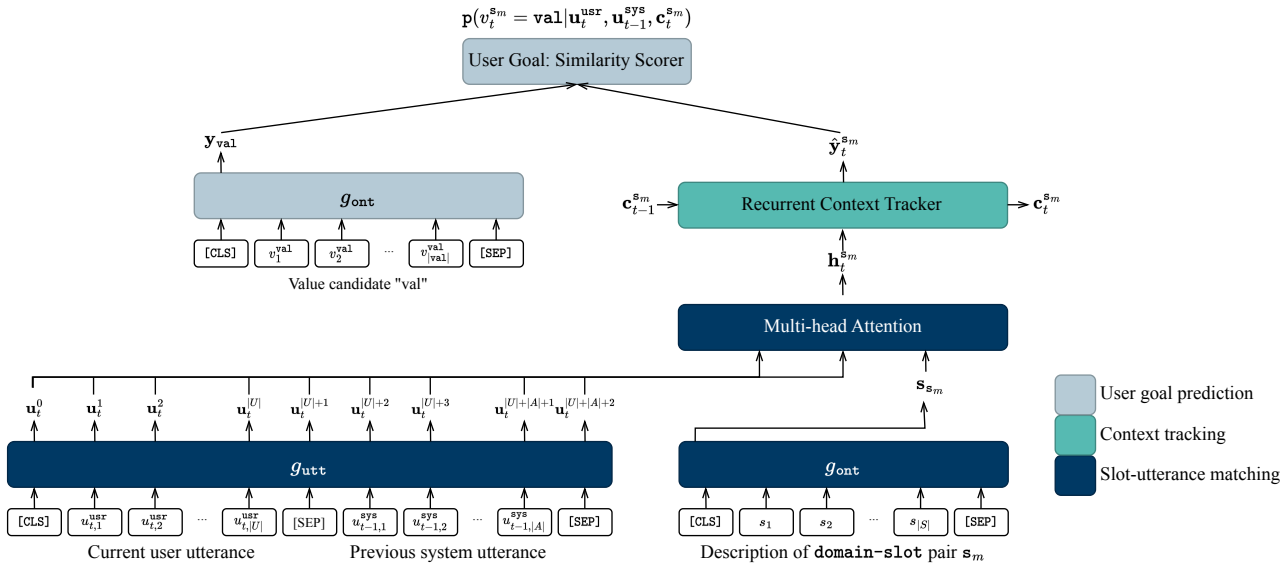


FIGURE 4.3: Slot utterance matching belief tracker.

4.3.2 Utterance and Ontology Features

At the core of the SUMBT model is the transformation of utterances and ontology terms into a shared representation space. The SUMBT model achieves this through contextual word embeddings

produced by a encoder language model, like BERT (Devlin et al., 2019). It employs two independent encoders, namely, the utterance encoder, $g_{\texttt{utt}}(\cdot)$ and ontology term encoder, $g_{\texttt{ont}}(\cdot)$ (as shown in dark blue in Figure 4.3). The utterance encoder is fine-tuned during the model training process to extract and learn essential features of conversational language. Conversely, the ontology encoder remains fixed during training, ensuring the generalisability of the model to unseen values.

To construct the current turn's representations, the system utterance is concatenated to the user response. The utterance encoder generates contextual representations, $\boldsymbol{U}_t$ from the utterance pair.

$$\boldsymbol{U}_t = g_{\texttt{utt}}\left(\left[\texttt{[CLS]} \quad \boldsymbol{u}_t^{\texttt{usr}} \quad \texttt{[SEP]} \quad \boldsymbol{u}_{t-1}^{\texttt{sys}} \quad \texttt{[SEP]}\right]\right).$$

Here the `[CLS]` and `[SEP]` tokens are the special beginning of sequence and end of sequence tokens used in the BERT model (see Section 2.11.4 for more information on such tokens). Similarly, the ontology encoder encodes the `domain-slot` pairs, and `values` independently, producing the pooled representations $\boldsymbol{s}_{\texttt{s}_m}$ and $\boldsymbol{y}_{\texttt{val}}$. Each term's representation is derived using the `[CLS]` token representation, which provides a comprehensive representation of the term.

$$\begin{aligned}\boldsymbol{s}_{\texttt{s}_m} &= \boldsymbol{h}_{\texttt{[CLS]}} = g_{\texttt{ont}}\left(\left[\texttt{[CLS]} \quad \texttt{domain-slot pair s}_m \quad \texttt{[SEP]}\right]\right), \text{ and}\\ \boldsymbol{y}_{\texttt{val}} &= \boldsymbol{h}_{\texttt{[CLS]}} = g_{\texttt{ont}}\left(\left[\texttt{[CLS]} \quad \texttt{value} \quad \texttt{[SEP]}\right]\right).\end{aligned}$$

4.3.3 Slot Utterance Matching

The slot-utterance matching component learns to associate relevant information in the utterances with specific domain-slot pairs through a multi-head attention mechanism, similar to that used for encoder-decoder attention by Vaswani et al. (2017). It uses the domain-slot representation as a query and the utterance representations as both the keys and values, producing a turn level representation for `domain-slot` pair $\texttt{s}_m$:

$$\boldsymbol{h}_t^{\texttt{s}_m} = g_{\texttt{MHA}}\left(\boldsymbol{U}_t, \boldsymbol{U}_t, \boldsymbol{s}_{\texttt{s}_m}\right),$$

where $g_{\texttt{MHA}}$ is a multi-head attention layer. This method allows the model to capture complex dependencies between different parts of the utterance and relate them to the appropriate domain-slot pair. The shared parameters across all domain-slot pairs from the ontology provide scalability to the model and allow it to generalise across different slots. For more information on the multi-head attention mechanism and the meanings of queries, keys and values see Section 2.8.

4.3.4 Context Tracking

The context tracking component of the SUMBT model maintains the conversational context across multiple turns using a latent state updated by a recurrent neural network (RNN). This latent state, $\boldsymbol{c}_t^{\texttt{s}_m}$, is updated based on the previous latent state $\boldsymbol{c}_{t-1}^{\texttt{s}_m}$ and the current slot-utterance information $\boldsymbol{h}_t^{\texttt{s}_m}$:

$$\boldsymbol{c}_t^{\texttt{s}_m} = f_{\texttt{RNN}}\left(\boldsymbol{h}_t^{\texttt{s}_m}, \boldsymbol{c}_{t-1}^{\texttt{s}_m}\right).$$

The use of a shared RNN across all domain-slot pairs, enables the model to keep track of the context in a scalable manner. Lee et al. (2019) found that a gated recurrent network (GRU) type RNN model performed the best in this setting.

4.3.5 User Goal Prediction

In the SUMBT model, the user goal, represented as a collection of `domain-slot-value` triplets, is predicted using a matching network approach (Vinyals et al., 2016). The matching network computes the semantic similarity between a value candidate representations, $\boldsymbol{y}_{\texttt{val}} \in \mathcal{V}_{\texttt{s}_m}$, and the current latent state, $\boldsymbol{c}_t^{\texttt{s}_m}$.

First, a linear feed-forward network is applied to the latent state $\boldsymbol{c}_t^{\mathsf{s}_m}$. The output of this transformation is then normalised, forming the predicted value representation $\hat{\boldsymbol{y}}_t^{\mathsf{s}_m}$. The resulting matching network scoring is expressed as:

$$z_t^{\mathsf{s}_m,\mathtt{val}} = \psi(\hat{\boldsymbol{y}}_t^{\mathsf{s}_m}, \boldsymbol{y}_{\mathtt{val}}),$$

where $\psi(\cdot,\cdot)$ denotes a measure of similarity between two vectors. In this context, cosine similarity is used, indicated by $\psi_{\cos}$.

The softmax function is applied to the scores for different value candidates to generate a belief state distribution:

$$\mathsf{p}\left(\mathsf{V}_t^{\mathsf{s}_m} = v \middle| \boldsymbol{u}_t^{\mathsf{usr}}, \boldsymbol{u}_{t-1}^{\mathsf{sys}}, \boldsymbol{c}_t^{\mathsf{s}_m}\right) = \frac{\exp\left(z_t^{\mathsf{s}_m,v}\right)}{\sum_{v' \in \mathcal{V}_{\mathsf{s}_m}} \exp\left(z_t^{\mathsf{s}_m,v'}\right)}.$$

Unlike traditional classification layers that predict labels based on learned weights, the similarity-based classification layer used here computes the cosine similarity between the transformed latent state and value candidate representations, making it flexible enough to score unseen values during training.

In the SUMBT model, the observed user action o_t is represented by the word-level user utterance $\boldsymbol{u}_t^{\mathsf{usr}}$ and the system action a_{t-1} by the word-level system utterance $\boldsymbol{u}_{t-1}^{\mathsf{sys}}$. The latent state $\boldsymbol{c}_t^{\mathsf{s}_m}$ is assumed to satisfy the Markov assumption. The belief state, therefore, is a concatenation of the distributions over values for each domain-slot pair in the model ontology. Hence, we can rewrite the belief state as:

$$\mathsf{b}\left(\boldsymbol{s}_t\right) = \left[\mathsf{p}\left(\mathsf{V}_t^{\mathsf{s}_m} \middle| \boldsymbol{u}_t^{\mathsf{usr}}, \boldsymbol{u}_{t-1}^{\mathsf{sys}}, \boldsymbol{c}_t^{\mathsf{s}_m}\right)\right]_{\forall \mathsf{s}_m \in \mathcal{O}},$$

where $\boldsymbol{s}_t$ is the complete dialogue state and c a domain-slot pair in the model ontology.

4.3.6 Training Objective

The objective is to generate the correct dialogue state by minimising the distance between the actual `value`, $v_t^{\mathsf{s}_m}$, and the latent state features, $\hat{\boldsymbol{y}}_t^{\mathsf{s}_m}$. This is achieved by maximising the likelihood over all turns t for every `domain-slot` pair s_m. This objective can be expressed as:

$$\mathcal{L}\left(\mathcal{D};\boldsymbol{\theta}\right) = -\frac{1}{D}\sum_{i=1}^{D}\sum_{m=1}^{M}\sum_{t=1}^{T_i}\log \mathsf{p}\left(\mathsf{V}_t^{s^m} = v_{i,t}^{s^m} | \boldsymbol{u}_t^{\mathsf{usr}}, \boldsymbol{u}_{t-1}^{\mathsf{sys}}, \boldsymbol{c}_t^{s^m}\right),$$

In the above equation, $\boldsymbol{\theta}$ represents the set of parameters of the model, T_i corresponds to the number of turns, and $\mathsf{V}_t^{\mathsf{s}_m}$ denotes the true value for domain, slot pair s_m at turn t. The loss is averaged across all dialogues in a batch during training.

4.3.7 Evaluation

In order to evaluate dialogue belief tracking models, such as SUMBT, we introduce two primary metrics metrics: (1) joint goal accuracy (JGA) and (2) expected calibration error (ECE). Joint goal accuracy measures the accuracy of the dialogue states predicted by the model and expected calibration error measures how well calibrated the uncertainty estimates of the model are. Before we introduce these models let us define the dialogue belief tracking model $f_{\mathsf{DBT}}(\cdot|\boldsymbol{\theta})$, parameterised by $\boldsymbol{\theta}$. For a given dialogue this model predicts the distribution over possible values for each `domain-slot` pair s_m:

$$\mathsf{p}\left(\mathsf{V}_t^{\mathsf{s}_m} \middle| \boldsymbol{u}_t^{\mathsf{usr}}, \boldsymbol{u}_{t-1}^{\mathsf{sys}}, \boldsymbol{c}_{t-1}^{\mathsf{s}_m}\right) = f_{\mathsf{DBT}}\left(\boldsymbol{u}_t^{\mathsf{usr}}, \boldsymbol{u}_{t-1}^{\mathsf{sys}}, \boldsymbol{c}_{t-1}^{\mathsf{s}_m}; \boldsymbol{\theta}\right).$$

Given a set of evaluation dialogues $\mathcal{D}_{\texttt{eval}} = \left\{ \left(\boldsymbol{u}^{\texttt{usr}}_{t,i}, \boldsymbol{u}^{\texttt{sys}}_{t-1,i}, \mathcal{B}_{t,i} \right)_{t=1}^{T_i} \right\}_{i=1}^{D_{\texttt{eval}}}$, the joint goal accuracy is defined as:

$$\texttt{JGA}\left(\mathcal{D}_{\texttt{eval}}; \boldsymbol{\theta}\right) = \frac{1}{\sum_{i=1}^{D_{\texttt{eval}}} T_i} \sum_{i=1}^{D_{\texttt{eval}}} \sum_{t=1}^{T_i} \prod_{m=1}^{M} \delta\left(\texttt{p}\left(\texttt{V}_t^{\texttt{s}_m} \middle| \boldsymbol{u}^{\texttt{usr}}_{t,i}, \boldsymbol{u}^{\texttt{sys}}_{t-1,i}, \boldsymbol{c}^{\texttt{s}_m}_{t-1,i}\right), v^{\texttt{s}_m}_{t,i}\right),$$

where:

$$\delta\left(\texttt{p}\left(\texttt{V}\right), v\right) = \begin{cases} 1 & \text{if } \arg\max'_v \texttt{p}\left(\texttt{V} = v'\right) = v \\ 0 & \text{otherwise} \end{cases}.$$

This is called the joint goal accuracy as the model is required to make a correct prediction for every `domain-slot` pair, the joint goal, in order for its prediction to count as correct.

Secondly the expected calibration error (ECE) measures how well calibrated a model is. This is done by measuring the expected error in calibration, i.e. the difference between the model confidence and the empirical likelihood of the model's predictions. In this context joint goal accuracy is used to measure the emperical likelihood of a model's predictions. ECE is mathematically represented by:

$$\texttt{ECE}\left(\mathcal{D}_{\texttt{eval}}; \boldsymbol{\theta}\right) = \sum_{b=1}^{B} \frac{|\mathbb{B}_b|}{D_{\texttt{eval}}} \left| \texttt{C}\left(\mathbb{B}_b\right) - \texttt{JGA}\left(\mathbb{B}_b\right) \right|.$$

Here $\mathbb{B}_b$ is the set of dialogue turns that fall into bin b, these dialogue turns are sorted by their confidences and split into B bins. $\texttt{C}\left(\mathbb{B}_b\right)$ is the average confidence of the model for dialogue turns in bin $\mathbb{B}_b$.

4.4 Conclusion

This chapter explored the evolution of Dialogue State Tracking (DST) models, their role in task-oriented dialogue systems, and their transformation from generative models to the present-day large language models.

Initially, dialogue modelling was envisioned as Markov Decision Processes (MDPs) but faced limitations due to the unobserved nature of the dialogue state. To tackle this, statistical dialogue models introduced the belief state, leading to the development of partially observable MDPs (POMDPs). Despite the improvement, these models grappled with the computational complexity arising from a large state space. This prompted the development of discriminative models, which modelled the conditional distribution of the state and bypassed the need for a vast state space.

The move towards integrated approaches streamlined natural language understanding and state tracking, extracting dialogue state directly from word-level features. Advancements in word vector representations have been crucial, moving from count-based methods to learned vector representation-based methods, and then to contextualised embeddings. Recently, large language models have significantly influenced DST, with generative models, span-prediction models, and picklist style classification models leading the advancements.

Several important areas have been identified for further exploration and improvement in Dialogue State Tracking (DST) models:

Handling Uncertainty and Calibration: The overconfidence of models in their predictions is a major issue that needs addressing. Future work could focus on designing mechanisms to generate belief state distributions that better mirror the actual uncertainty in the predicted states, leading to a more accurate and reliable model performance.

Ontology Feature Representation: Current methods often rely on the beginning of sequence (`[BOS]` or `[CLS]`) token to encapsulate the representation of each term in the model ontology. However,

this approach might overlook the semantic richness in multi-word terms. Future DST models could benefit from richer and more comprehensive embeddings based on term descriptions or sequences of embeddings, enhancing their understanding of nuanced terms and thereby improving their overall performance.

Efficient Data Usage: DST models currently demand large quantities of annotations due to the need for value annotation for each domain-slot pair at every dialogue turn. Span-prediction models require additional data for span annotations. Future research could aim to devise methods to mitigate these data requirements or enhance the efficiency of data usage.

Evaluation Metrics: Present evaluation metrics such as Joint Goal Accuracy may not sufficiently capture the entirety of model performance, particularly in terms of calibration. The development of more comprehensive and nuanced metrics is essential for a holistic evaluation of model performance. Furthermore, evaluations should also measure the downstream impact of DST models on the dialogue system's policy since the primary objective of DST models is to equip the policy with all the necessary information for effective decision making.

In summary, the pursuit of the optimal DST model remains a continuous process. This endeavour is propelled by ongoing advancements in NLP, machine learning techniques, and the escalating complexity of real-world dialogue systems. These areas of improvement signify the future pathways to refine DST models and enrich the dialogue systems they support.

Chapter 5

Knowing What You Know: Calibrating Dialogue Belief State Distributions via Ensembles

This chapter summarises our work on the calibration of dialogue belief state distributions using ensembles and gives a verbatim copy of our paper (van Niekerk et al., 2020):

Carel van Niekerk et al. (2020). "Knowing What You Know: Calibrating Dialogue Belief State Distributions via Ensembles". In: *Findings of the Association for Computational Linguistics: EMNLP 2020*. Online: Association for Computational Linguistics, pp. 3096–3102. DOI: 10.18653/v1/2020.findings-emnlp.277. URL: https://www.aclweb.org/anthology/2020.findings-emnlp.277

5.1 Summary

This work introduces the first approach to address the calibration of dialogue belief state distributions. The proposed method, CE-SUMBT, combines the label smoothed maximum likelihood objective with an ensemble of models to generate well-calibrated belief state distributions. CE-SUMBT achieves state-of-the-art calibration performance on the MultiWOZ 2.1 dataset (Eric et al., 2020). Furthermore, the study demonstrates that in a well-calibrated belief state, the correct dialogue state candidate ranks among the top three candidates in 84% of turns, indicating that a well-calibrated belief state potentially provides informative information for decision-making.

5.2 Personal Contributions

The implementation and technical results are my own work, while my co-authors contributed to the writing and proofreading process.

Knowing What You Know: Calibrating Dialogue Belief State Distributions via Ensembles

Carel van Niekerk, Michael Heck, Christian Geishauser
Hsien-Chin Lin, Nurul Lubis, Marco Moresi, Milica Gašić
Heinrich Heine University Düsseldorf, Germany
niekerk,heckmi,geishaus,linh,lubis,moresi,gasic@hhu.de

Abstract

The ability to accurately track what happens during a conversation is essential for the performance of a dialogue system. Current state-of-the-art multi-domain dialogue state trackers achieve just over 55% accuracy on the current go-to benchmark, which means that in almost every second dialogue turn they place full confidence in an incorrect dialogue state. Belief trackers, on the other hand, maintain a distribution over possible dialogue states. However, they lack in performance compared to dialogue state trackers, and do not produce well calibrated distributions. In this work we present state-of-the-art performance in calibration for multi-domain dialogue belief trackers using a calibrated ensemble of models. Our resulting dialogue belief tracker also outperforms previous dialogue belief tracking models in terms of accuracy.

1 Introduction

Task-oriented dialogue systems aim to act as assistants to their users, solving tasks such as finding a restaurant, booking a train, or providing information about a tourist attraction. They have become very popular with the introduction of virtual assistants such as Siri and Alexa.

Two tasks are fundamental to such a system. The first is the ability to track what happened in the conversation, referred to as **tracking**. Based on the result of tracking, the system needs to conduct the conversation towards the fulfilment of the user goal, referred to as **planning**. The tracking component summarises the dialogue history, or the past, while the planning component manages the dialogue and concerns the future. In this work we focus on the first component.

Early approaches to statistical dialogue modelling view dialogue as a Markov decision process (Levin et al., 1998) and define a set of dialogue states that the conversation can be in at any given dialogue turn. The tracking component tracks the **dialogue state**. In recent years discriminative models achieve state-of-the-art dialogue state tracking (DST) results (Kim et al., 2019; Zhang et al., 2019; Heck et al., 2020). Still, in a multi-domain setting such as MultiWOZ (Eric et al., 2019; Budzianowski et al., 2018), they achieve an accuracy of just over 55%. This means that in approximately 45% of cases they make a wrong prediction and, even worse, they have full confidence in that wrong prediction.

In the wake of statistical dialogue modeling, the use of partially observable Markov decision processes has been proposed to address this issue. The idea is to model the probability over all possible dialogue states in every dialogue turn (Williams and Young, 2007). This probability distribution is referred to as the **belief state**. The advantages of belief tracking are probably best illustrated by an excerpt from a dialogue with a real user in (Metallinou et al., 2013): even though the dialogue state predicted with the highest probability is not the true one, the system is able to provide a valid response because the true dialogue state also has assigned a non-zero probability.

> A model is considered well **calibrated** if its confidence estimates are aligned with the empirical likelihood of its predictions (Desai and Durrett, 2020).

The belief state can be modelled by deep learning-based approaches such as the neural belief tracker (Mrkšić et al., 2017), the multi-domain belief tracker (Ramadan et al., 2018), the globally conditioned encoder belief tracker (Nouri and Hosseini-Asl, 2018) and the slot utterance matching belief tracker (SUMBT) (Lee et al., 2019) models. None of these models however address the issue of calibrating the probability distribution that

they provide, resulting in them being more confident than they should be. In a dialogue setting, overconfidence can lead to bad decisions and unsuccessful dialogues.

In this work, we present methods for learning well-calibrated belief distributions. Our contributions are the following:

- We present the state-of-the-art performance in calibration for dialogue belief trackers using a calibrated ensemble of models, called the calibrated ensemble belief state tracker (CEBST).
- Our model achieves best overall joint goal accuracy among the state-of-the-art **belief** tracking models.

Such a well-calibrated belief tracking model is essential for the planning component to successfully conduct dialogue.

2 Related Work

Since no other belief tracking methods that we are aware of have achieved success in producing well-calibrated confidence, we look towards methods used in other language tasks. Natural language inference is a related task that also benefits from well-calibrated confidence in predictions. Desai and Durrett (2020) introduce the use of post-processing techniques such as temperature scaling to produce better-calibrated confidence estimates.

Additionally, there have been recent advances in the construction of more adequate loss functions. These methods, including Bayesian matching and prior networks, aim to learn well-calibrated models without the burden of requiring many extra parameters. These methods achieve good calibration in computer vision tasks such as CIFAR (Joo et al., 2020; Malinin and Gales, 2018; Szegedy et al., 2016).

When the limitations of a single model still inhibit us from producing more accurate and better-calibrated models, a popular alternative is to use an ensemble of models. Recently Malinin and Gales (2020) showed the success of using an ensemble of models for machine translation, and in particular utilising accurate confidence predictions for analysing translation quality.

3 Calibration Techniques

In this section we explain the details of three calibration techniques that we apply to dialogue belief tracking.

3.1 Loss Functions

The loss function can have a great impact on the calibration and accuracy of models. The most commonly used loss function in belief tracking is the standard softmax cross entropy loss. However, it tends to cause overconfident predictions where most of the probability is placed on the top class.

Label smoothing cross entropy (Szegedy et al., 2016) aims to resolve this problem by replacing the one-hot targets of cross entropy with a smoothed target distribution. That is, for label y_i and smoothing parameter $\alpha \in \left(0, \frac{1}{K}\right]$, the target distribution will be:

$$t(c|\alpha, y_i) = \begin{cases} 1-(K-1)\alpha & c = y_i, \\ \alpha & \text{otherwise}, \end{cases} \quad (1)$$

where K is the number of possible values of c. The loss for a model with parameters $\boldsymbol{\theta}$ and a set of N output logits $\hat{\mathbf{z}}_1, \hat{\mathbf{z}}_2, ..., \hat{\mathbf{z}}_N$ with true labels $y_1, y_2, ..., y_N$ is defined as:

$$\mathcal{L}(\boldsymbol{\theta}, \alpha) = \frac{1}{N}\sum_{i=1}^{N} \mathbf{KL}\left[\text{Softmax}(\hat{\mathbf{z}}_i)||t(c_i|\alpha, y_i)\right], \quad (2)$$

where $\mathbf{KL}$ is the Kullback–Leibler divergence between two distributions (Kullback and Leibler, 1951).

Alternatively, Bayesian matching loss (Joo et al., 2020) uses a Dirichlet distribution as the final activation function. The target is constructed using the Bayes rule, where we assume the observed label y_i to be an observation from a categorical distribution $y_i|\boldsymbol{\pi}_i \sim \text{Cat}(\boldsymbol{\pi}_i)$ and $\boldsymbol{\pi}_i$ is the true underlying distribution of the label. To introduce uncertainty into the target distribution we assume that the prior of $\boldsymbol{\pi}_i$ is a Dirichlet distribution, $\text{Dir}(\mathbf{1})$. In this way, we have a highly uncertain prior distribution. From this it can be shown that the posterior will be $\boldsymbol{\pi}_i|y_i \sim \text{Dir}(\mathbf{1} + \mathbf{I}(y_i))$, where $\mathbf{I}(y_i)$ is the one-hot representation of y_i. The loss function is then constructed using the negative log likelihood of the true label given the predicted distribution $\hat{\boldsymbol{\pi}}_i \sim \text{Dir}(\hat{\mathbf{z}}_i)$, penalised by the KL divergence from the the uncertain $\text{Dir}(\mathbf{1})$ distribution:

$$\mathcal{L}(\boldsymbol{\theta}, \lambda) = \sum_{i=1}^{N}\{\lambda \mathbf{KL}\left[\hat{\boldsymbol{\pi}}_i||\text{Dir}(\mathbf{1})\right] - \mathbb{E}_{\hat{\boldsymbol{\pi}}_i}\left[\log(p(y_i|\hat{\boldsymbol{\pi}}_i))\right]\}, \quad (3)$$

where $\lambda > 0$ is the penalisation parameter.

3.2 Ensemble Distribution Estimation

From a Bayesian viewpoint, the probability of observing an outcome given the observed examples can be broken down into two components: the predictive distribution of the model and the posterior of the model given the observed examples. The posterior of the model given the data is an unknown distribution which can be estimated in various ways. One method is to use an ensemble of models, where the ensemble acts as an estimator for the posterior distribution of the parameters, $p(\boldsymbol{\theta}|\mathcal{D})$, where $\mathcal{D}$ represents the observed examples. Let $q(\boldsymbol{\theta})$ represent the distribution over all possible members of an ensemble. This distribution could be seen as the ensemble estimate of the posterior, $p(\boldsymbol{\theta}|\mathcal{D})$, (Malinin et al., 2019; Malinin and Gales, 2020). Hence,

$$\hat{p}(y|\mathbf{x}, \mathcal{D}) = \int p(y|\mathbf{x}, \boldsymbol{\theta}) q(\boldsymbol{\theta}) \mathrm{d}\boldsymbol{\theta}. \qquad (4)$$

Since this integral is still intractable we need to estimate it using Monte Carlo. To sample from the ensemble distribution $q(\boldsymbol{\theta})$ we consider two approaches: using dropout during inference to collect an ensemble of N equally likely models (Gal and Ghahramani, 2016), or alternatively bootstrap sampling N equally likely subsets of the data to train N equally likely ensemble members. Let these N members be $\{\boldsymbol{\theta}^{(1)}, \boldsymbol{\theta}^{(2)}, ..., \boldsymbol{\theta}^{(N)}\}$. The estimated predictive distribution can then be calculated as follows:

$$\hat{p}(y|\mathbf{x}, \mathcal{D}) = \frac{1}{N} \sum_{i=1}^{N} p(y|\mathbf{x}, \boldsymbol{\theta}^{(i)}) \qquad (5)$$

3.3 Temperature Scaling

Temperature scaling is a post-processing technique which scales the logits of the model by a scaling factor $\beta > 1$ (Guo et al., 2017), resulting in better-calibrated estimates. The temperature scaling parameter β can be trained on a development set.

4 Experimental Setup

We seek to build a well-calibrated dialogue belief tracker. For our baseline belief tracker, we use the SUMBT model architecture (Lee et al., 2019), which uses BERT (Devlin et al., 2018) as a turn encoder and multi-head attention for slot candidate matching. We perform all experiments on the MultiWOZ 2.1 dataset (Eric et al., 2019), the current standard dataset for multi-domain dialogue. When training using Bayesian matching, we use a scaling coefficient of $\lambda = 0.003$, and for label smoothing, a smoothing coefficient of $\alpha = 0.05$. For the ensemble belief tracker, we train 10 identical independent models, each with a sub-sample of 7500 dialogues. All hyper-parameters are obtained using a parameter search based on validation set performance. For all training, we use the BERT-base-uncased model from PyTorch Transformers (Wolf et al., 2019) for turn embedding. We use a gated recurrent unit with a hidden dimension 300 for latent tracking and Euclidean distance for value candidate scoring. During training, we use a learning rate of $5e-5$ in combination with a linear learning rate scheduler, the warm-up proportion is set to 0.1. A dropout rate of 0.3 is used, and training is performed for 100 epochs.[1]

5 Evaluation Metrics

5.1 Joint Goal Accuracy

The joint goal accuracy (JGA) is the percentage of turns for which the model predicts the complete user goal correctly. We further propose the introduction of an adjusted top 3 JGA, which considers a user goal prediction correct if the true label for each slot is among the top 3 predicted candidates for that slot in the belief state given there are at least 5 possible candidates.

5.2 L2 Norm Error

The L2 norm error is the L2 norm of the difference between the true labels and the predicted distributions. To form the user goals and belief states we concatenate all the slot labels and slot distributions. This error measure does not only consider the accuracy of the predictions but also the uncertainty.

5.3 Joint Goal Calibration Error

A well-calibrated model is one where the accuracy is aligned with the confidence predictions. The expected calibration error (ECE) evaluates the calibration by measuring the difference between the model's confidence and accuracy (Guo et al., 2017), meaning a lower ECE indicates better calibration. Hence:

$$\mathrm{ECE} = \sum_{k=1}^{B} \frac{b_k}{N} |\mathrm{acc}(k) - \mathrm{conf}(k)|, \qquad (6)$$

[1] Our code will be made available at https://gitlab.cs.uni-duesseldorf.de/general/dsml/calibrating-dialogue-belief-state-distributions.

where B is the number of bins, b_k are the bin sizes, N the number of observations, $\mathrm{acc}(k)$ and $\mathrm{conf}(k)$ the accuracy and confidence measures of bin k. We also propose an adapted ECE, called the expected joint goal calibration error (EJCE), which uses the joint goal accuracy for bin k as $\mathrm{acc}(k)$, and the following metric as confidence:

$$\mathrm{conf(k)} = \frac{1}{b_k} \sum_{i=1}^{b_k} \min_{s \in \text{slots}} \max_{v \in \text{values}} \hat{p}_i(v|s), \quad (7)$$

where $\hat{p}_i(v|s)$ is the predicted probability of value v for slot s given the i^{th} observation in bin k.

6 Results

Model	JGA	Top 3 JGA	EJCE
Cross entropy	46.78%	69.97%	1.996
Label smoothing	46.32%	74.57%	1.292
Bayesian matching	31.03%	45.16%	4.922
Temperature scaling			
Cross entropy (1.73*)	46.78%	69.97%	4.758
Label smoothing (1.00*)	46.32%	74.57%	1.292
Dropout ensembles			
Cross entropy (35**)	47.18%	71.14%	2.909
Label smoothing (35**)	46.36%	76.12%	2.217
Bootstrap model ensembles			
Label smoothing (10**)	**48.41%**	**84.08%**	**0.841**

Table 1: Calibration strategy performance. *temperature scaling coefficient **ensemble size.

Model	JGA	L2 Norm
SUMBT (Lee et al., 2019)	46.78%	1.1075
CE-BST (ours)	48.41%	**1.1041**
SOTA DST	$<$ **56.0%**	$>$ 1.2445

Table 2: MultiWOZ 2.1 performance.

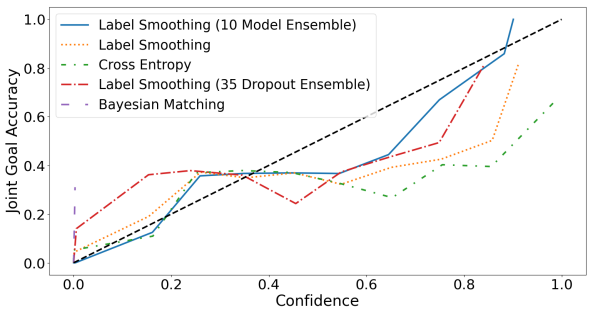


Figure 1: Reliability Diagram.

All of the calibration techniques presented above can be combined. Here, we focus on the most important combinations and present the results in Table 1. We make the following observations. First, cross entropy on its own leads to a high EJCE, as expected. Second, label smoothing reduces EJCE while leading to a negligible drop in accuracy. Third, Bayesian matching underperformed in our experiments, suggesting a difficulty in choosing the right priors. Fourth, temperature scaling is not an effective way of calibrating uncertainty, as the same calibration is applied to each observation. Finally, the ensemble methods produce very promising results for both accuracy and calibration of the model. In particular, if we look at the Top 3 JGA, our method achieves an improvement of 14.11 percentage points over the baseline, in the Appendix we include a comprehensive set of Top n JGA results. In Figure 1 we plot JGA as a function of confidence. The best calibrated model is the one that is closest to the diagonal, i.e. the one whose confidence for each dialogue state is closest to the achieved accuracy. From this reliability diagram we see that both the dropout and model ensembles improve model calibration and do not produce over-confident output as the cross entropy baseline does. In Table 2 we compare our model to some of the best performing belief and state tracking models. Here we see that we outperform the best performing **belief** tracker but the state-of-the-art (SOTA) **state** trackers (Heck et al., 2020; Chen et al., 2020; Hosseini-Asl et al., 2020) have a significantly higher JGA. However, when analysing the L2 norm[2] we see that the uncertainty estimates of **belief** tracking models compensate for the lower joint goal accuracy. This corroborates our premise that it is important to have well calibrated confidence estimates and not just a high JGA.

7 Conclusion

We applied a number of calibration techniques to a baseline dialogue belief tracker. We showed that a label smoothed trained ensemble provides state-of-the-art calibration of the belief state distributions and has the best accuracy among the available **belief** trackers. Although it does not compete with **state** trackers in terms of JGA, when considering top 3 predictions it achieves 84.08% accuracy

[2]For a model with a given JGA we can calculate the minimum L2 that such a model can possibly achieve by assuming that it never predicts more than one slot incorrectly.

(Top 3 JGA), almost 30 percentage points above state-of-the art state trackers. We also find that our model has the best L2 norm performance, which suggests that the quality of predicted uncertainty is as important as the average JGA.

It is important to note that the proposed calibration methods can be applied to any neural dialogue belief tracking method. The uncertainty estimates predicted by this model could improve the success of dialogue systems because this model can provide the dialogue manager with a good measure of confidence. This could allow the system to ask questions in moments of confusion. In the Appendix we include example dialogues to illustrate this. In future, we aim to combine the state-of-the-art dialogue state tracking and belief tracking methods to create a method that can achieve both states-of-the-art joint goal accuracy and well-calibrated belief states.

Acknowledgements

C. van Niekerk, M. Heck and N. Lubis are supported by funding provided by the Alexander von Humboldt Foundation in the framework of the Sofja Kovalevskaja Award endowed by the Federal Ministry of Education and Research, while C. Geishauser, H-C. Lin and M. Moresi are supported by funds from the European Research Council (ERC) provided under the Horizon 2020 research and innovation programme (Grant agreement No. STG2018 804636).

References

Paweł Budzianowski, Tsung-Hsien Wen, Bo-Hsiang Tseng, Iñigo Casanueva, Ultes Stefan, Ramadan Osman, and Milica Gašić. 2018. MultiWOZ - a large-scale multi-domain Wizard-of-Oz dataset for task-oriented dialogue modelling. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Lu Chen, Boer Lv, Chi Wang, Su Zhu, Bowen Tan, and Kai Yu. 2020. Schema-guided multi-domain dialogue state tracking with graph attention neural networks. *AAAI 2020*.

Shrey Desai and Greg Durrett. 2020. Calibration of pre-trained Transformers. *arXiv preprint arXiv:2003.07892*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Mihail Eric, Rahul Goel, Shachi Paul, Abhishek Sethi, Sanchit Agarwal, Shuyag Gao, and Dilek Hakkani-Tur. 2019. MultiWOZ 2.1: Multi-domain dialogue state corrections and state tracking baselines. *arXiv preprint arXiv:1907.01669*.

Y Gal and Z Ghahramani. 2016. Dropout as a Bayesian approximation: Representing model uncertainty in deep learning. In *33rd International Conference on Machine Learning, ICML 2016*, volume 3, pages 1651–1660.

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. 2017. On calibration of modern neural networks. In *International Conference on Machine Learning*, pages 1321–1330.

Michael Heck, Carel van Niekerk, Nurul Lubis, Christian Geishauser, Hsien-Chin Lin, Marco Moresi, and Milica Gasic. 2020. TripPy: A triple copy strategy for value independent neural dialog state tracking. In *Proceedings of the 21th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 35–44, 1st virtual meeting. Association for Computational Linguistics.

Ehsan Hosseini-Asl, Bryan McCann, Chien-Sheng Wu, Semih Yavuz, and Richard Socher. 2020. A simple language model for task-oriented dialogue. *arXiv preprint arXiv:2005.00796*.

Taejong Joo, Uijung Chung, and Min-Gwan Seo. 2020. Being Bayesian about categorical probability. *arXiv preprint arXiv:2002.07965*.

Sungdong Kim, Sohee Yang, Gyuwan Kim, and Sang-Woo Lee. 2019. Efficient dialogue state tracking by selectively overwriting memory. *arXiv preprint arXiv:1911.03906*.

Solomon Kullback and Richard A Leibler. 1951. On information and sufficiency. *The annals of mathematical statistics*, 22(1):79–86.

Hwaran Lee, Jinsik Lee, and Tae-Yoon Kim. 2019. SUMBT: slot-utterance matching for universal and scalable belief tracking. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5478–5483.

Esther Levin, Roberto Pieraccini, and Wieland Eckert. 1998. Using Markov decision process for learning dialogue strategies. In *Proceedings of the 1998 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP'98 (Cat. No. 98CH36181)*, volume 1, pages 201–204. IEEE.

Andrey Malinin and Mark Gales. 2018. Predictive uncertainty estimation via prior networks. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pages 7047–7058.

Andrey Malinin and Mark Gales. 2020. Uncertainty in structured prediction. *arXiv preprint arXiv:2002.07650*.

Andrey Malinin, Bruno Mlodozeniec, and Mark Gales. 2019. Ensemble distribution distillation. *arXiv preprint arXiv:1905.00076*.

Angeliki Metallinou, Dan Bohus, and Jason D. Williams. 2013. Discriminative state tracking for spoken dialog systems. In *Proceedings of Annual Meeting of the Association for Computational Linguistics (ACL), Sofia, Bulgaria*. Association for Computational Linguistics.

Nikola Mrkšić, Diarmuid Ó Séaghdha, Tsung-Hsien Wen, Blaise Thomson, and Steve Young. 2017. Neural belief tracker: Data-driven dialogue state tracking. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1777–1788, Vancouver, Canada. Association for Computational Linguistics.

Elnaz Nouri and Ehsan Hosseini-Asl. 2018. Toward scalable neural dialogue state tracking model. *arXiv preprint arXiv:1812.00899*.

Osman Ramadan, Paweł Budzianowski, and Milica Gašić. 2018. Large-scale multi-domain belief tracking with knowledge sharing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 432–437.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. 2016. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826.

Jason D Williams and Steve Young. 2007. Partially observable Markov decision processes for spoken dialog systems. *Computer Speech & Language*, 21(2):393–422.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, R'emi Louf, Morgan Funtowicz, and Jamie Brew. 2019. Huggingface's transformers: State-of-the-art natural language processing. *ArXiv*, abs/1910.03771.

Jian-Guo Zhang, Kazuma Hashimoto, Chien-Sheng Wu, Yao Wan, Philip S Yu, Richard Socher, and Caiming Xiong. 2019. Find or classify? Dual strategy for slot-value predictions on multi-domain dialog state tracking. *arXiv preprint arXiv:1910.03544*.

A Appendices

A.1 Joint Goal Accuracy Analysis

In Table 3 we compare SUMBT and our CE-BST method using 5 different top n joint goal accuracy's.

Model	Top 1	Top 2	Top 3	Top 4	Top 5
SUMBT	46.78%	64.61%	69.97%	72.10%	73.70%
CE-BST	48.41%	77.25%	84.08%	85.84%	86.93%

Table 3: Top n joint goal accuracy comparison.

A.2 Example Dialogues

In Figures 2 - 9 we present some example dialogues together with an extract from their belief state distributions. These examples show situations where a well-calibrated belief state distribution could be beneficial for decision making.

User: I need a place to stay.
System: Sure. I'll need a little more information. Is there an area you are interested in?
User: No specific area. I would like it to be in the moderate price range and it should have free parking. I would also like it to have 4 stars.

Figure 2: Dialogue *PMUL3364* from the MultiWOZ 2.1 corpus.

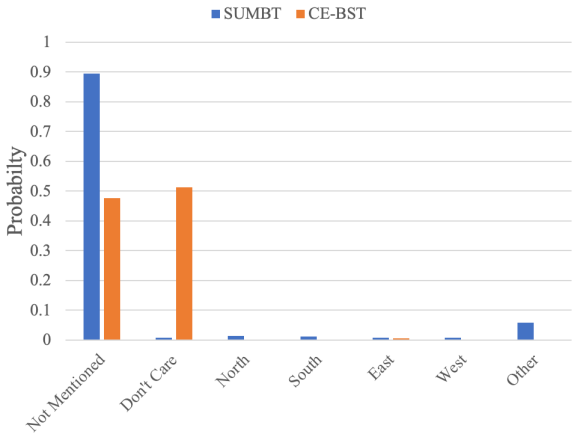


Figure 3: *PMUL3364* Hotel - Location belief state distribution.

User: Can you help me find a place to go in the centre?
System: I can help you with that. Is there a certain kind of attraction that you would like to visit?
User: Surprise me! Give me the postcode as well.
System: Would you prefer the castle galleries is a museum in the centre of town. Their post code is cb23bj.
User: Great! I am also looking for a place to eat in the same area. Something not too expensive, but not cheap.

Figure 4: Dialogue *PMUL4258* from the MultiWOZ 2.1 corpus.

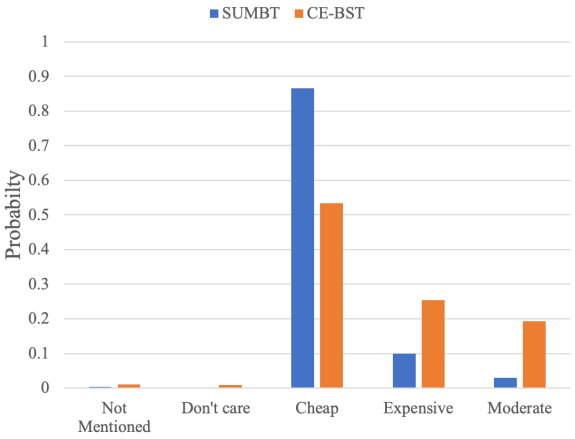


Figure 5: *PMUL4258* Restaurant - Price Range belief state distribution.

User: Hi, I am looking for a hotel by the name of Acorn guest house.
System: Sure, what would you like to know about it?
User: I would like to know if it is available for 8 people for 4 nights starting Saturday?

Figure 6: Dialogue *PMUL4605* from the MultiWOZ 2.1.

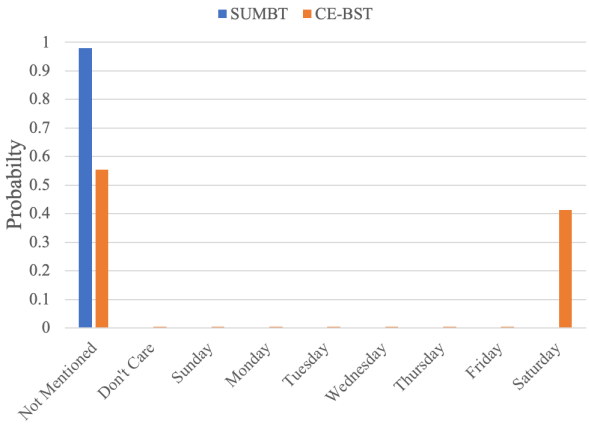


Figure 7: *PMUL4605* Hotel - Book Day belief state distribution.

User: I need a train to Stansted airport that leaves on Sunday.
System: Did you have a time you would like to arrive or leave?
User: I need to arrive by 14:30.
System: tr1668 will arrive at 14:08, would that work for you?
User: That is perfect. I would like to make a booking for 6 people please.
System: Booking was successful, the total fee is 48.48 gbp payable at the station. Your reference number is hf03ug02. Do you need assistance with anything else?
User: I need to eat too.
System: What type of restaurant and price range are you looking for?
User: I would like Catalan food. It needs to be in the centre and be expensive.

Figure 8: Dialogue *PMUL3625* from the MultiWOZ 2.1 corpus.

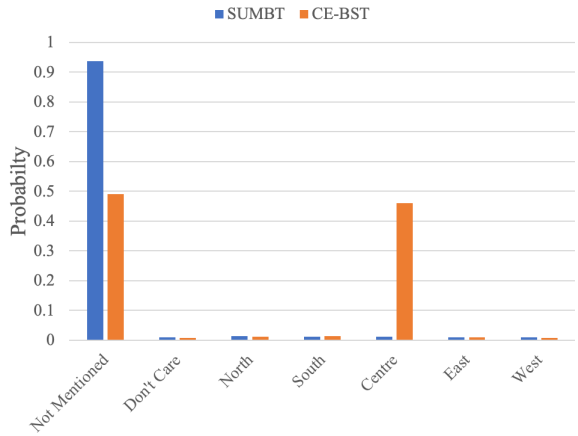


Figure 9: *PMUL3625* Restaurant - Location belief state distribution.

Chapter 6

Uncertainty Measures in Neural Belief Tracking and the Effects on Dialogue Policy Performance

This chapter summarises our work on uncertainty estimation in dialogue belief tracking and the downstream impact on the dialogue policy module and gives a verbatim copy of our paper (van Niekerk et al., 2021):

Carel van Niekerk et al. (2021). "Uncertainty Measures in Neural Belief Tracking and the Effects on Dialogue Policy Performance". In: *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. Online and Punta Cana, Dominican Republic: Association for Computational Linguistics. URL: https://aclanthology.org/2021.emnlp-main.623

6.1 Summary

In our previous work van Niekerk et al. (2020), in Chapter 5, we introduced a calibrated ensemble neural belief tracking framework. This framework not only allowed neural belief trackers to produce well-calibrated uncertainty measures but also estimate knowledge uncertainty, which arises when the model lacks sufficient training data to make an accurate prediction. While they may be useful for crafting robust dialogue policy models, the ensemble framework is computationally demanding during inference, hindering real-time responses. Moreover, these belief tracking models only track the user goal component of a dialogue state, not addressing all the requirements of a Markov dialogue state (Young et al., 2010).

In this paper, we propose an advanced neural belief tracking model employing a novel set-based similarity approach. The model, in addition to the user goal, predicts critical components of the dialogue state, such as the user request and current active domains. Through the application of ensemble distribution distillation (EnD^2) (Ryabinin et al., 2021), we distil the ensemble model, resulting in a well-calibrated model capable of real-time predictions. A human-trial affirms that incorporating knowledge uncertainty significantly boosts policy robustness against noise compared to policies that neglect uncertainty. These findings underline both the importance of uncertainty estimation in neural belief tracking models and the potential of policy models to use this uncertainty to resolve conversational ambiguities.

6.2 Personal Contributions

The implementation and technical results are my own work, while my co-authors contributed to the writing and proofreading process.

Uncertainty Measures in Neural Belief Tracking and the Effects on Dialogue Policy Performance

Carel van Niekerk[1], **Andrey Malinin**[2], **Christian Geishauser**[1], **Michael Heck**[1]
Hsien-chin Lin[1], **Nurul Lubis**[1], **Shutong Feng**[1] **and Milica Gašić**[1]
[1]Heinrich Heine Universität Düsseldorf, Düsseldorf, Germany
[2]Yandex Research and HSE University, Moscow, Russia
[1]{niekerk,geishaus,heckmi,linh,lubis,shutong.feng,gasic}@hhu.de
[2]am969@yandex-team.ru

Abstract

The ability to identify and resolve uncertainty is crucial for the robustness of a dialogue system. Indeed, this has been confirmed empirically on systems that utilise Bayesian approaches to dialogue belief tracking. However, such systems consider only confidence estimates and have difficulty scaling to more complex settings. Neural dialogue systems, on the other hand, rarely take uncertainties into account. They are therefore overconfident in their decisions and less robust. Moreover, the performance of the tracking task is often evaluated in isolation, without consideration of its effect on the downstream policy optimisation. We propose the use of different uncertainty measures in neural belief tracking. The effects of these measures on the downstream task of policy optimisation are evaluated by adding selected measures of uncertainty to the feature space of the policy and training policies through interaction with a user simulator. Both human and simulated user results show that incorporating these measures leads to improvements both of the performance and of the robustness of the downstream dialogue policy. This highlights the importance of developing neural dialogue belief trackers that take uncertainty into account.

1 Introduction

In task-oriented dialogue, the system aims to assist the user in obtaining information. This is achieved through a series of interactions between the user and the system. As the conversation progresses, it is the role of the *dialogue state tracking* module to track the state of the conversation. For example, in a restaurant recommendation system, the state would include the information about the cuisine of the desired restaurant, its area as well as the price range that the user has in mind. It is crucial that this state contains all information necessary for the *dialogue policy* to make an informed decision for the next action (Young et al., 2007). Policy training optimises decision making in order to complete dialogues successfully.

It has been proposed within the partially observable Markov decision process (POMDP) approach to dialogue modelling to track the distribution over all possible dialogue states, the *belief state*, instead of a single most-likely candidate. This approach successfully integrates uncertainty to achieve robustness (Williams and Young, 2007; Thomson and Young, 2010; Young et al., 2016, 2007). However, such systems do not scale well to complex multi-domain dialogues. On the other hand, discriminative neural approaches to dialogue tracking achieve state-of-the-art performance in the state tracking task. Nevertheless, the state-of-the-art goal accuracy on the popular MultiWOZ (Budzianowski et al., 2018) multi-domain benchmark is currently only at 60% (Heck et al., 2020; Li et al., 2020a). In other words, even the best neural dialogue state trackers at present incorrectly predict the state of the conversation in 40% of the turns. What is particularly problematic is that these models are fully confident about their incorrect predictions.

Unlike neural dialogue state trackers, which predict a single best dialogue state, neural belief trackers produce a belief state (Williams and Young, 2007; Henderson et al., 2013). State-of-the-art neural belief trackers, however, achieve an even lower goal accuracy of approximately 50% (van Niekerk et al., 2020; Lee et al., 2019), making the more accurate state trackers a preferred approach. High-performing state trackers typically rely on span-prediction approaches, which are unable to produce a distribution over all possible states as they extract information directly from the dialogue.

Ensembles of models are known to yield improved predictive performance as well as a calibrated and rich set of uncertainty estimates (Malinin, 2019; Gal, 2016). Unfortunately, ensemble generation and, especially, inference come at a high computational and memory cost which may be pro-

Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, pages 7901–7914
November 7–11, 2021.

hibitive. While standard ensemble distillation (Hinton et al., 2015) can be used to compress an ensemble into a single model, information about ensemble diversity, and therefore several uncertainty measures, is lost. Recently Malinin et al. (2019) and Ryabinin et al. (2021) proposed *ensemble distribution distillation* (EnD^2) - an approach to distill an ensemble into a single model which preserves both the ensemble's improved performance and full set of uncertainty measures at low inference cost.

In this work we use EnD^2 to distill an ensemble of neural belief trackers into a single model and incorporate additional uncertainty measures, namely confidence scores, total uncertainty (entropy) and knowledge uncertainty (mutual information), into the belief state of the neural dialogue system. This yields an uncertainty-aware neural belief tracker and allows downstream dialogue policy models to use this information to resolve confusion. To our knowledge, ensemble distillation, especially ensemble distribution distillation, and the derived uncertainty estimates, have not been examined for belief state estimation or *any* downstream tasks.

We make the following contributions:

1. We present SetSUMBT, a modified SUMBT belief tracking model, which incorporates set similarity for accurate state predictions and produces components essential for policy optimisation.
2. We deploy ensemble distribution distillation to obtain well-calibrated, rich estimates of uncertainty in the dialogue belief tracker. The resulting model produces state-of-the-art results in terms of calibration measures.
3. We demonstrate the effect of adding additional uncertainty measures in the belief state on the downstream dialogue policy models and confirm the effectiveness of these measures both in a simulated environment and in a human trial.

2 Background

2.1 Dialogue Belief Tracking

In statistical approaches to dialogue, one can view the dialogue as a Markov decision process (MDP) (Levin et al., 1998). This MDP maintains a Markov *dialogue state* in each turn and chooses its next *action* based on this state.

Alternatively, we can model the dialogue state as a latent variable, maintaining a *belief state* at each turn, as in partially observable Markov decision processes (POMDPs) (Williams and Young, 2007; Thomson and Young, 2010). While attractive in theory, the POMDP model is computationally expensive in practice. Although there are practical implementations, they are limited to single-domain dialogues and their performance fall short of discriminative statistical belief trackers (Williams, 2012). The inherent problem lies in the generative nature of POMDP trackers where the state generates noisy observations. This becomes an issue for instance when the user wants to change the goal of a conversation, e.g., the user wants an Italian instead of a French restaurant. Henderson (2015) has shown empirically that discriminative models model a change in user goal more accurately.

In discriminative approaches, the state depends on the observation, making it easier for the system to identify a change of the user goal. Traditional discriminative approaches suffer from low robustness, as they depend on static semantic dictionaries for feature extraction (Henderson et al., 2014; Mrkšić et al., 2017b). Integrated approaches on the other hand utilise learned token vector representations, leading to more robust state trackers (Mrkšić et al., 2017a; Ramadan et al., 2018; Lee et al., 2019). However, highly over-parameterised models, such as neural networks – when trained via maximum-likelihood on finite data – often yield miscalibrated, over-confident predictions, placing *all* probability mass on a single outcome (Pleiss et al., 2017). Consequently, belief tracking is reduced to state tracking, losing the benefits of uncertainty management. State-of-the-art approaches to dialogue state tracking redefine the problem as a span-prediction task. These models extract the values directly from the dialogue context (Chao and Lane, 2019; Zhang et al., 2020; Heck et al., 2020) and manage to achieve state-of-the-art results on MultiWOZ (Budzianowski et al., 2018; Eric et al., 2020). Span-prediction models at present do not produce probability distributions, so additional work is needed to apply our proposed uncertainty framework to them. Neural belief and state trackers rarely model the correlation between domain-slot pairs, except for works by Hu et al. (2020) and Ye et al. (2021). Due to scalability issues we do not include these approaches in our investigation. We therefore consider the slot-utterance matching belief tracker (SUMBT) (Lee et al., 2019) a better starting point, as it is readily able to produce a belief state distribution.

In theory, well-calibrated belief trackers have an inherent advantage over state tracking, producing uncertainty estimates that lead to more robust downstream policy performance. This raises the question: Is it possible to instil well-calibrated uncertainty estimates in neural belief trackers? And if so, do these estimates have a positive effect on the downstream policy optimisation in practice?

We believe SUMBT is a fitting approach to investigate these questions, as it has been shown that an ensemble of SUMBT models can achieve state-of-the-art goal L2-Error when trained using specialised loss functions aiming at inducing uncertainty in the output (van Niekerk et al., 2020).

2.2 Ensemble-based Uncertainty Estimation

Consider a classification problem with a set of features $\boldsymbol{x}$, and outcomes $y \in \{\omega_1, \omega_2, ..., \omega_K\}$. In dialogue state tracking, $\boldsymbol{x}$ would be features of the input to the tracker and y would be a dialogue state. Given an ensemble of M models $\left\{\mathtt{P}(y|\boldsymbol{x}, \boldsymbol{\theta}^{(m)})\right\}_{m=1}^{M}$, the *predictive posterior* is obtained as follows:

$$\mathtt{P}(y|\boldsymbol{x}, \mathcal{D}) = \sum_{m=1}^{M} \frac{\mathtt{P}(y|\boldsymbol{x}, \boldsymbol{\theta}^{(m)})}{M} =: \sum_{m=1}^{M} \frac{\boldsymbol{\pi}^{(m)}}{M} \quad (1)$$

Predictions made using the predictive posterior are often better than those of individual models. The entropy $\mathcal{H}[]$ of the predictive posterior is an estimate of *total uncertainty*. Ensembles allow decomposing *total uncertainty* into *data* and *knowledge uncertainty* by considering measures of *ensemble diversity*. *Data uncertainty* is the uncertainty due to noise, ambiguity and class overlap in the data. *Knowledge uncertainty* is uncertainty due to a lack of knowledge of the model about a *test data* (Malinin, 2019; Gal, 2016) — ie, uncertainty due to unfamiliar, anomalous or atypical inputs. Ideally, ensembles should yield *consistent* predictions on data similar to the training data and *diverse* predictions on data which is significantly different from the training data. Thus measures of ensemble diversity yield estimates of *knowledge uncertainty*[1]. These quantities are obtained via the mutual information $\mathcal{I}[y, \boldsymbol{\theta}]$ between predictions and model parameters. The quantity in the Equation 2 is a measure of *ensemble diversity*, and therefore, *knowledge uncertainty*. This quantity is the difference between the entropy of the predictive posterior (total uncertainty) and the average entropy of each model in the ensemble (data uncertainty).

$$\underbrace{\mathcal{I}[y, \boldsymbol{\theta}|\boldsymbol{x}, \mathcal{D}]}_{\text{Knowledge unc.}} = \underbrace{\mathcal{H}\left[\mathtt{P}(y|\boldsymbol{x}, \mathcal{D})\right]}_{\text{Total uncertainty}} - \underbrace{\sum_{m=1}^{M} \frac{\mathcal{H}\left[\mathtt{P}(y|\boldsymbol{x}, \boldsymbol{\theta}^{(m)})\right]}{M}}_{\text{Data uncertainty}} \quad (2)$$

2.3 Ensemble Distillation

While ensembles provide improved predictive performance and a rich set of uncertainty measures, their practical application is limited by their inference-time computational cost. Ensemble distillation (EnD) (Hinton et al., 2015) can be used to compress an ensemble into a single student model (with parameters $\boldsymbol{\phi}$) by minimising the Kullback-Leibler (KL) divergence between the ensemble predictive posterior and the distilled model predictive posterior, significantly reducing the inference cost. Unfortunately, a significant drawback of this method is that information about *ensemble diversity*, and therefore *knowledge uncertainty*, is lost in the process. Recently, Malinin et al. (2019) proposed *ensemble distribution distillation* (EnD^2) as an approach to distill an ensemble into a single *prior network* model (Malinin and Gales, 2018), such that the model retains information about ensemble diversity. Prior networks yield a higher-order *Dirichlet distribution* over categorical output distributions $\boldsymbol{\pi}$ and thereby *emulate* ensembles, whose output distributions can be seen as samples from a higher-order distribution[2]. Formally, a prior network is defined as follows:

$$\begin{aligned} &\mathtt{p}(\boldsymbol{\pi}|\boldsymbol{x}; \boldsymbol{\phi}) = \mathtt{Dir}(\boldsymbol{\pi}|\boldsymbol{\alpha}), \ \boldsymbol{\alpha} = e^{\boldsymbol{z}} \\ &\boldsymbol{z} = \boldsymbol{f}(\boldsymbol{x}; \boldsymbol{\phi}), \ \alpha_k > 0, \ \alpha_0 = \sum_{k=1}^{K} \alpha_k, \end{aligned} \quad (3)$$

where $\mathtt{Dir}(\cdot|\boldsymbol{\alpha})$ is a Dirichlet distribution with concentration parameters $\boldsymbol{\alpha}$, and $\boldsymbol{f}(\cdot; \boldsymbol{\phi})$ is a learned function which yields the logits $\boldsymbol{z}$. The predictive posterior can be obtained in closed form though marginalisation over $\boldsymbol{\pi}$, thereby emulating (1). This yields a softmax output function:

$$\begin{aligned} \mathtt{P}(y = \omega_k|\boldsymbol{x}; \boldsymbol{\phi}) &= \mathbb{E}_{\mathtt{p}(\boldsymbol{\pi}|\boldsymbol{x}; \boldsymbol{\phi})}[\mathtt{P}(y = \omega_k|\boldsymbol{\pi})] \\ &= \frac{e^{z_k}}{\sum_{k=1}^{K} e^{z_k}}. \end{aligned} \quad (4)$$

[1] In-depth overviews of ensemble methods are available in Malinin (2019); Gal and Ghahramani (2016); Ashukha et al. (2020); Ovadia et al. (2019).

[2] $\boldsymbol{\pi} = [\mathtt{P}(y = \omega_1|\boldsymbol{x}), \cdots, \mathtt{P}(y = \omega_K|\boldsymbol{x})]^{\mathrm{T}}$.

Closed form estimates of all uncertainty measures are obtained via Eq. (5), which emulates the same underlying mechanics as Eq. (2), as follows (Malinin, 2019):

$$\underbrace{\mathcal{I}[y, \boldsymbol{\pi}|\boldsymbol{x}; \boldsymbol{\phi}]}_{\text{Knowledge unc.}} = \underbrace{\mathcal{H}\left[\mathrm{P}(y|\boldsymbol{x}; \boldsymbol{\phi})\right]}_{\text{Total uncertainty}} - \underbrace{\mathbb{E}_{\mathrm{p}(\boldsymbol{\pi}|\boldsymbol{x};\boldsymbol{\phi})}\left[\mathcal{H}[\mathrm{P}(y|\boldsymbol{\pi})]\right]}_{\text{Data uncertainty}}. \quad (5)$$

Originally, Malinin et al. (2019) implemented EnD^2 on the CIFAR10, CIFAR100 and TinyImageNet datasets. However, Ryabinin et al. (2021) found scaling to tasks with many classes challenging using the original Dirichlet Negative log-likelihood criterion. They analysed this scaling problem and proposed to a new loss function, which minimises the *reverse KL-divergence* between the model and an intermediate *proxy Dirichlet* target derived from the ensemble. This loss function was shown to enable EnD^2 on tasks with arbitrary numbers of classes. In this work we use this improved loss function, as detailed in the Appendix Section B.2.

2.4 Policy Optimisation

In each turn of dialogue, the dialogue policy selects an action to take in order to successfully complete the dialogue. The input to the policy is constructed using the output of the belief state tracker, thus being directly impacted by its richness.

Optimising dialogue policies within the original POMDP framework is not practical for most cases. Therefore, the POMDP is viewed as a continuous MDP whose state space is the belief space. This state space can be discretised, so that tabular reinforcement learning (RL) algorithms can be applied (Gašić et al., 2008; Thomson et al., 2010). Gaussian process RL can be applied directly on the original belief space (Gašić and Young, 2014). This is also possible using neural approaches with less computational effort (Jurčíček et al., 2011; Weisz et al., 2018; Chen et al., 2020). Current state-of-the-art RL algorithms for multi-domain dialogue management (Takanobu et al., 2019; Li et al., 2020b) utilise proximal policy optimisation (Schulman et al., 2017) operating on single best dialogue state.

3 Effects of Uncertainty on Downstream Tasks

We take the following steps in order to examine the effects of the additional uncertainty measures in the dialogue belief state:

1. Modify the original SUMBT model (Lee et al., 2019) to arrive at a competitive baseline. We call this model SetSUMBT.
2. Produce ensembles of SetSUMBT following the work of van Niekerk et al. (2020).
3. Apply EnD and EnD^2 as introduced in Section 2.3.
4. Apply policy optimisation that uses belief states from distilled models.

3.1 Neural Belief Tracking Model

We propose a neural belief tracker which one can easily incorporate in a full dialogue system pipeline. We base our tracker on the slot-utterance matching belief tracker (SUMBT) (Lee et al., 2019), but we make two important changes. First, we ensure our tracker is fully in line with the requirements of the hidden information state (HIS) model for dialogue management (Young et al., 2007) by adding user action predictions to our tracker. These are not produced by the SUMBT model and nor by other available neural trackers. However, they are essential for integration into a full dialogue system. Second, in order to improve the understanding ability of the model, we utilise a set of concept description embeddings rather than a single embedding for semantic concepts. We use this set of embeddings for information extraction and prediction, hence we call our model SetSUMBT. In this section we describe each component in detail, also depicted in Figure 1.

Slot-utterance matching The slot-utterance matching (SUM) component performs the role of language understanding in the SUMBT architecture. The SUM multi-head attention mechanism (Vaswani et al., 2017) attends to the relevant information in the current turn for a specific domain-slot pair. In the process of slot-utterance matching, SUMBT utilises BERT's (Devlin et al., 2019) `[CLS]` sequence embedding to represent the semantic concepts in the model ontology. Instead of using the single `[CLS]` embedding, we make use of the sequence of embeddings for the domain-slot description. We choose to make this expansion, as approaches which utilise a sequence of embeddings outperform approaches based on a single embedding in various natural language processing tasks (Poerner et al., 2020; Choi et al., 2021). We further use RoBERTa as a feature extractor (Liu et al., 2019).

Dialogue context tracking The first of the three components of the HIS model is a representation of the dialogue context (history). In the SUMBT approach, a gated-recurrent unit mechanism tracks the most important information during a dialogue. The resulting context conditioned representations for the domain-slot pairs contain the relevant information from the dialogue history. Similar to the alteration in the SUM component, we represent the dialogue context as a sequence of representations. This sequence, $\boldsymbol{C}_t^s$, represents the dialogue context for domain-slot pair s across turns 1 to t, while it dimension being independent of t. Besides the above modification, we add a further step where we reduce this sequence of context representations to a single representation $\hat{\mathbf{y}}_t^s$. We do this reduction using a learned convolutional pooler, which we call the *Set Pooler*. See Appendix Section C for more details regarding the implementation.

User goal prediction The second component of the HIS model is the user goal. This is typically the only component that neural tracing models explicitly model as a set of domain-slot-value pairs. Here, we follow the matching network approach (Vinyals et al., 2016) utilised by SUMBT, where the predictive distribution is based on the similarity between the dialogue context and the value candidates. To obtain the similarity between the dialogue context and a value candidate we make use of cosine similarity, $S_{\cos}(\cdot,\cdot)$. Based on these similarity scores, we produce a predictive distribution, Equation 6, for the value of domain-slot pair s at turn t v_t^s, the user and system utterances at turn t, $\boldsymbol{u}_t^{usr}$ and $\boldsymbol{u}_{t-1}^{sys}$, and the dialogue context representations at turn $t-1$ $\boldsymbol{C}_{t-1}^s$. Contrary to the SUMBT approach, each value candidate is represented by the sequence of value description embeddings from a fixed RoBERTa model. The *Set Pooler*, with the same parameters used for pooling context representations, reduces this sequence of value description representations to a representation $\mathbf{y}_v$, for value v.

$$\mathtt{P}\left(v_t^s = v|\boldsymbol{u}_t^{usr}, \boldsymbol{u}_{t-1}^{sys}, \boldsymbol{C}_{t-1}^s\right) = \frac{\exp\left(S_{\cos}\left(\hat{\mathbf{y}}_t^s, \mathbf{y}_v\right)\right)}{\sum_{v'}\exp\left(S_{\cos}\left(\hat{\mathbf{y}}_t^s, \mathbf{y}_{v'}\right)\right)}, \tag{6}$$

User action prediction To be fully in line with the HIS model, we further require the predicted user actions. In order to predict the user actions, we categorise them into general user actions and user request actions. Further, since our system is a multi-domain system, we include the current active domain in the hidden information state of the system.

General user action includes actions such as the user thanking or greeting the system, which do not rely on the dialogue context. Hence, we can infer general user actions from the current user utterance. A user request action is an action indicating that the user is requesting information about an entity. Zhu et al. (2020) shows that simple rule-based estimates of these actions lead to poor downstream policy performance. Hence, we propose predicting this information within the belief tracking model.

Since we can infer the general actions from the current user utterance, we use a single turn representation $\boldsymbol{x}_t^0$ to predict such actions. The single turn representation, $\boldsymbol{x}_t^0$, is the representation for the RoBERTa sequence representation `<s>`, which is equivalent to the BERT `[CLS]` representation. That is:

$$\begin{aligned}\mathtt{P}\left(a_t^{gen} = a|\mathbf{u}_t^{usr}, \mathbf{u}_{t-1}^{sys}\right) = \\ \text{softmax}\left(\boldsymbol{W}^{gen}\mathbf{x}_t^0 + \boldsymbol{b}^{gen}\right),\end{aligned} \tag{7}$$

where $a \in \{\texttt{none}, \texttt{thank\_you}, \texttt{goodbye}\}$.

The more difficult sub-tasks include active user request and active domain prediction. For user request prediction we utilise the dialogue context representation $\hat{\mathbf{y}}_t^s$ for a specific domain-slot pair to predict whether the user has requested information relating to this slot. That is:

$$\begin{aligned}\mathtt{P}\left(r_t^s = 1|\boldsymbol{u}_t^{usr}, \boldsymbol{u}_{t-1}^{sys}, \boldsymbol{C}_{t-1}^s\right) = \\ \text{sigmoid}\left(\boldsymbol{w}^{req}\hat{\mathbf{y}}_t^s + b^{req}\right),\end{aligned} \tag{8}$$

where r_t^s indicates an active request for domain-slot s by the user in turn t.

Last, to predict active domains in the dialogue, we incorporate information relating to all slots associated with a specific domain. We do so by performing mean reduction across the context representations of all the slots associated with a domain. The resulting domain representations are used to predict whether a domain is currently being discussed in the dialogue. That is, for active domain d_t, S_d the set of slots within domain d, and $\boldsymbol{C}_{t-1}^d := \left[\boldsymbol{C}_{t-1}^s\right]_{s\in S_d}$ the set of context representations for all domain-slot pairs in S_d at turn $t-1$, we have the active domain distribution:

$$\begin{aligned}\mathtt{P}\left(d_t = d|\boldsymbol{u}_t^{usr}, \boldsymbol{u}_{t-1}^{sys}, \boldsymbol{C}_{t-1}^d\right) = \\ \text{sigmoid}\left(\boldsymbol{w}^{dom}\frac{1}{|S_d|}\sum_{s\in S_d}\hat{\mathbf{y}}_t^s + b^{dom}\right),\end{aligned} \tag{9}$$

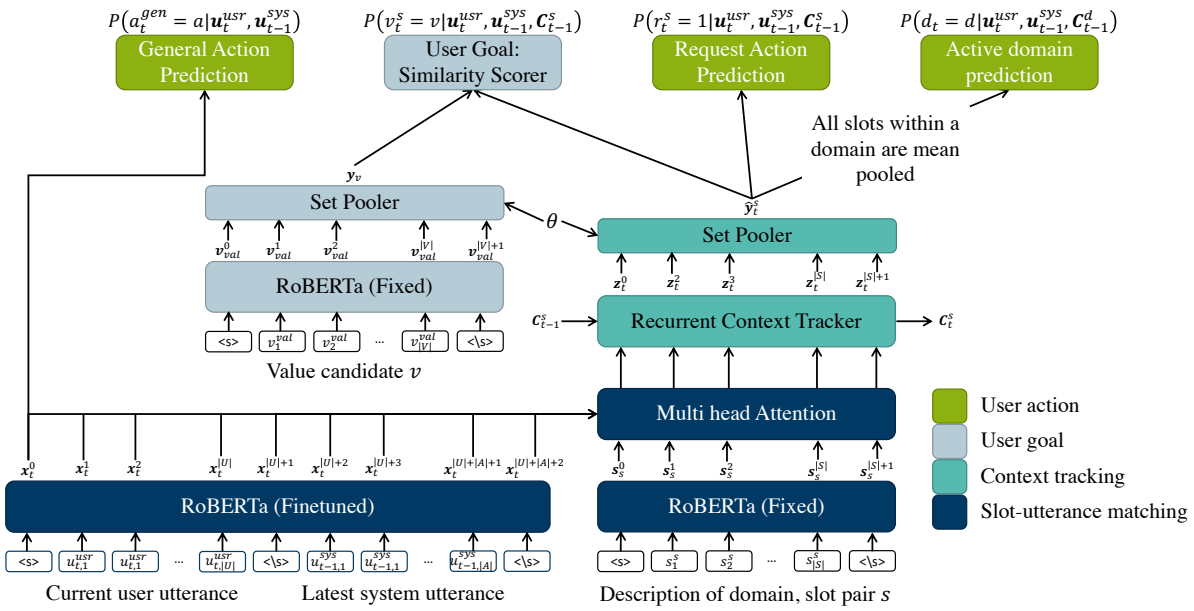


Figure 1: Architecture of our SetSUMBT model, which takes as input the current user utterance, the latest system utterance, and a domain-slot pair description. The model, further, requires a pre-defined set of plausible value candidates for each domain-slot pair. At each turn, we encode the utterances only once, the *Slot-utterance matching* and *Context tracking* components are utilised once for each domain-slot pair. Further, we use the *Set Pooler* once for each domain-slot pair and once for each value candidate. The *Set Pooler* used for pooling value candidate and domain-slot context sequences shares the same parameters $\boldsymbol{\theta}$. SetSUMBT outputs a belief state distribution for the relevant domain-slot pair (*User goal*), a distribution over general actions, and the probability of a user request for the domain-slot pair (*User action*). The model also outputs the probability of an active domain.

Optimisation For each of the four tasks: user goal prediction, general user action prediction, user request action prediction and active domain prediction, the aim of the model is to predict the correct class. To optimise for these objectives, we minimise the following classification loss functions: $\mathcal{L}_{\text{goal}}$, $\mathcal{L}_{\text{general}}$, $\mathcal{L}_{\text{request}}$ and $\mathcal{L}_{\text{domain}}$. During model training we combine four weighted classification objectives:

$$\begin{aligned}\mathcal{L} = \alpha_{\text{goal}}\mathcal{L}_{\text{goal}} + \alpha_{\text{general}}\mathcal{L}_{\text{general}} \\ +\alpha_{\text{request}}\mathcal{L}_{\text{request}} + \alpha_{\text{domain}}\mathcal{L}_{\text{domain}},\end{aligned} \tag{10}$$

where $\alpha_x \in (0, 1]$ is the importance of task x. In this work, we use the label smoothing classification loss for all sub-tasks as it results in better calibrated predictions, as shown by van Niekerk et al. (2020), see details in Section B.1 of the appendix.

3.2 Uncertainty Estimation in SetSUMBT

Similarly to van Niekerk et al. (2020), we construct an ensemble of SetSUMBT models by training each model on one of 10 randomly selected subsets of data. We then distil this ensemble into a single model by adopting ensemble distillation (EnD) and ensemble distribution distillation (EnD^2) as described in Section 2.3. We refer to these distilled versions of the SetSUMBT ensemble as EnD-SetSUMBT and EnD^2-SetSUMBT, respectively.

The SetSUMBT belief tracker tracks the presence and value of each domain-slot pair s as the dialogue progresses. For the sake of scalability of the downstream policy, in the user goal g we do not consider all possible values, but rather the most likely one v^s for every domain-slot pair s and its associated probability, i.e., the confidence score given by $h^g_{t,s}$ summarised in vector $\mathbf{h}^g_t$ for all domain-slot pairs:

$$\begin{aligned} v^s &:= \arg\max_v \mathrm{P}\left(v^s_t = v | \boldsymbol{u}^{usr}_t, \boldsymbol{u}^{sys}_{t-1}, \boldsymbol{C}^s_{t-1}\right), \\ h^g_{t,s} &:= \max_v \mathrm{P}\left(v^s_t = v | \boldsymbol{u}^{usr}_t, \boldsymbol{u}^{sys}_{t-1}, \boldsymbol{C}^s_{t-1}\right), \\ \mathbf{h}^g_t &:= [v^s, h^g_{t,s}]_{\forall s}. \end{aligned} \tag{11}$$

For the EnD-SetSUMBT belief tracker, we can also calculate the total uncertainty for each domain-slot given by the entropy, see Section 2.3. We encode that information in $h^{unc}_{t,s}$ for each domain-slot pair

s and summarise in $\mathbf{h}_t^{unc}$ for all domain-slot pairs:

$$h_{t,s}^{unc} \coloneqq \mathcal{H}\left[\mathrm{P}\left(v_t^s = v | \boldsymbol{u}_t^{usr}, \boldsymbol{u}_{t-1}^{sys}, \boldsymbol{C}_{t-1}^s\right)\right],$$
$$\mathbf{h}_t^{unc} \coloneqq [h_{t,s}^{unc}]_{\forall s}.$$

For the EnD^2-SetSUMBT belief tracker, can further include the knowledge uncertainty for each domain-slot pair s given by the mutual information:

$$h_{t,s}^{unc} \coloneqq \mathcal{I}[v_t^s, \boldsymbol{\pi} | \boldsymbol{u}_t^{usr}, \boldsymbol{u}_{t-1}^{sys}, \boldsymbol{C}_{t-1}^s; \boldsymbol{\phi}],$$

as per Eq. (5) where $\boldsymbol{\pi}$ represents the ensemble distribution and $\boldsymbol{\phi}$ the model parameters.

In addition, all versions of SetSUMBT include the following vectors/variables:
$\mathbf{h}_t^g$ is the estimate of the user goal from Eq. (11),
$\mathbf{h}_t^{usr}$ is the estimate of user actions from Eq. (7-9),
$\mathbf{h}_t^{db}$ is the database search result[3],
$\mathbf{h}_{t-1}^{sys}$ is the system action,
$\mathbf{h}_t^{book}$ is the set of completed bookings,
h_t^{term} indicates the termination of the dialogue.

This results in the following belief state:

$$\mathbf{b}_t = \{\mathbf{h}_t^{usr}, \mathbf{h}_{t-1}^{sys}, \mathbf{h}_t^g, \mathbf{h}_t^{book}, \mathbf{h}_t^{db}, h_t^{term}, \mathbf{h}_t^{unc}\}.$$

For a system without uncertainty, all confidences would be rounded to either 0 or 1 and the belief state would not contain the $\mathbf{h}_t^{unc}$ vector.

3.3 Policy Optimisation as Downstream Task

For our experiments we optimise the dialogue policy operating on the belief state via RL using the PPO algorithm (Schulman et al., 2017). PPO is an on-policy actor-critic algorithm that is widely applied across different reinforcement learning tasks because of its good performance and simplicity. Similarly to Takanobu et al. (2019), we use supervised learning to pretrain the policy before starting the RL training phase. In order to perform supervised learning we need to map the belief states into system actions as they occur in the corpus. These belief states can either be oracle states taken from the corpus or predictions of our belief tracker that takes corpus dialogues as input. We investigate both options for policy training.

[3]Uncertainty is incorporated in the database query vector in the form of confidence thresholds. If the confidence score for a specific constraint is less than a chance prediction then this constraint is ignored during the database query.

Approach	JGA(%)	L2-Error	ECE(%)
SUMBT	46.78	1.1075	25.46
CE-BST	48.71	**1.1042**	10.73
SUMBT+LaRL	51.52	-	-
SetSUMBT	51.11	1.2386	15.13
EnD^2-SetSUMBT	51.22	1.1948	7.09
CE-SetSUMBT	52.04	1.1936	**6.84**
EnD-SetSUMBT	**52.26**	1.1782	7.54

Table 1: Comparison of neural belief tracking approaches on the MultiWOZ 2.1 test set. CE is an ensemble of calibrated models, EnD is ensemble distillation and EnD^2 is ensemble distribution distillation.

4 Experiments

4.1 Neural Belief Tracking Performance

Overall performance Table 1 compares the performance of our proposed SetSUMBT belief tracker to existing approaches, which include SUMBT, the calibrated ensemble belief state tracker (CE-BST) (van Niekerk et al., 2020) and the end to end trained SUMBT+LaRL approach (Lee et al., 2020). We consider the joint goal accuracy (JGA), L2-Error and expected calibration error (ECE). The JGA of a belief tracking model is the percentage of turns for which the model correctly predicted the value for all domain-slot pairs. The L2-Error is the L2-Norm of the difference between the predicted user distribution and the true user goal. Further, the ECE is the average absolute difference between the accuracy and the confidence of a model. In this comparison, we do not consider state tracking approaches, as they do not yield uncertainty estimates. SetSUMBT outperforms SUMBT and SUMBT+LaRL in terms of calibration and accuracy. We name the variants of SetSUMBT as follows: CE-SetSUMBT is a calibrated ensemble of SetSUMBT similar to CE-BST, EnD-SetSUMBT is the distilled SetSUMBT model, and EnD^2-SetSUMBT is the distribution distilled SetSUMBT model.

Runtime efficiency The single instance of the SetSUMBT tracker processes a dialogue turn in approximately 77.768 ms, whereas an ensemble of 10 models processes a turn in approximately 768.025 ms. These processing times are averaged across the 7372 turns in the MultiWOZ test set, see Appendix Section E for more details. The significant increase in processing time for the ensemble of models makes this approach inappropriate for real time interaction with users on a private device.

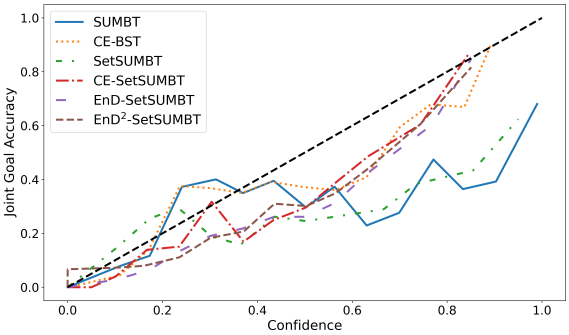


Figure 2: Reliability comparison of a selection of neural belief tracking models.

Belief Tracker	Belief state uncertainty	Success Rate	Reward	Turns
SetSUMBT	None	78.67	46.51	7.49
	Confidence	83.25	52.49	6.80
EnD-SetSUMBT	None	82.25	51.18	7.52
	Confidence	83.75	54.04	**6.46**
	Total	**86.83**	**57.09**	7.35
EnD^2-SetSUMBT	None	83.75	53.00	7.50
	Confidence	84.08	53.15	7.74
	Total	84.83	54.54	7.26
	Knowledge	85.17	54.63	7.57

Table 2: Performance of the systems in the simulated environment. For each setting we have 5 policies initiated with different random seeds, each evaluated with 1000 dialogues and their success rates, reward and number of turns averaged.

Calibration The reliability diagram in Figure 2 illustrates the relationship between the joint goal accuracy and the model confidence. The best calibrated model is the one that is closest to the diagonal, i.e., the one whose confidence for each dialogue state is closest to the achieved accuracy. The best reliability is achieved by CE-BST, and CE-SetSUMBT comes second. Both distillation models (EnD-SetSUMBT and EnD^2-SetSUMBT) do not deviate greatly from CE-SetSUMBT.

4.2 Policy Training on User Simulator

We incorporate SetSUMBT, EnD-SetSUMBT and EnD^2-SetSUMBT within the Convlab2 (Zhu et al., 2020) task-oriented dialogue environment and compare their performance by training policies which take belief states as inputs[4].

To investigate the impact of additional uncertainty measures on the dialogue policy we perform interactive learning in a more challenging environment than the original Convlab2 template-based simulator. We add ambiguity to the simulated user utterances in the form of value variations that occur in the MultiWOZ dataset. For example, instead of the user simulator asking for a hotel for "one person", it could also say "It will be just me.". For more information see Appendix Section D.

When policies are trained for large domains, they are typically first pretrained on the corpus in a supervised manner, and then improved using reinforcement learning. We first investigate which states to use for the supervised pretraining (Section 3.3): oracle states, i.e., the dialogue state labels from the MultiWOZ corpus, or estimated belief states, e.g., those predicted by a EnD-SetSUMBT model. We then evaluate the pretrained policies with the simulated user. During the evaluation both policies use a EnD-SetSUMBT model to provide belief states. We observe that the policy pretrained using the oracle state achieves a success rate of 36.50% in the simulated environment compared to the 46.08% success rate achieved by the policy pretrained using EnD-SetSUMBT. Thus, all our following experiments use predicted belief states of respective tracking models for the pretraining stage.

For each setting of the belief tracker we have four possible belief state settings, i.e., the binary state (no uncertainty), the confidence score state, the confidence score state with additional total uncertainty features and the confidence score state with additional knowledge uncertainty features. For each setting we evaluate the policies through interaction with the user simulator, results are given in Table 2.

In interaction with the simulator, systems making use of confidence outperform the systems without any uncertainty (significance at $p < 0.05$). Moreover, the additional total and knowledge uncertainty features always outperform the systems which only use a confidence score (significance at $p < 0.05$). This indicates that additional measures of uncertainty improve the robustness of the downstream dialogue policy in a challenging environment.

It is interesting to note that the system which makes use of total uncertainty appears to outperform the system that makes use of knowledge uncertainty (significance at $p < 0.05$). We suspect that this controlled simulated environment has low data uncertainty, so the total uncertainty is overall more informative.

[4] https://gitlab.cs.uni-duesseldorf.de/general/dsml/setsumbt-public.git

4.3 Human Trial

We conduct a human trial, where we compare SetSUMBT as the baseline with EnD-SetSUMBT and EnD^2-SetSUMBT. For EnD-SetSUMBT, we consider the model that includes both confidence scores and entropy features. For EnD^2-SetSUMBT, we investigate the model that includes confidence scores and knowledge uncertainty features. For each model we have two variations: one with a binary state corresponding to the most likely state (no uncertainty variation), and one with uncertainty measures (uncertainty variation). For each variation we chose the policy whose performance on the simulated user is closest to the average performance of its respective setting, see Section 4.2.

Subjects are recruited through the Amazon Mechanical Turk platform to interact with our systems via the DialCrowd platform (Lee et al., 2018). Each assignment consists of a dialogue task and two dialogues to perform. The task comprises a set of constraints and goals, for example finding the name and phone number of a guest house in the downtown area. We encourage the subjects to use variants of labels by introducing random value variants in the tasks. The two dialogues are performed in a random order with two variations of the same model, namely no-uncertainty and uncertainty variation, as described above. After each dialogue, the subject rates the system as successful if they think they received all the information required and all constraints were met. The subjects rate each system on a 5 point Likert scale. In total we collected approximately 550 dialogues for each of 6 different systems, 3300 in total. There was a total of 380 subjects who took part in these experiments.

Table 3 shows the performance of the above policies in the human trial. We confirm that each no uncertainty system is always worse than its uncertainty counterpart (each significant at $p < 0.05$). It is important to emphasise here that in each pairing, the systems have exactly the same JGA, but their final performance can be very different in terms of success and user rating. This empirically demonstrates the limitations of JGA as a single measure for dialogue state tracking, urging the modelling of uncertainty and utilisation of calibration measures. Finally, we observe that adding additional uncertainty measures improves the policy (each significant at $p < 0.05$) and the best overall performance is achieved by the system that utilises both knowledge uncertainty and confidence scores

Belief Tracker	Belief state uncertainty	Success Rate	Rating	Turns
SetSUMBT	None	48.99	2.68	7.28
	Confidence	67.05	3.47	**6.12**
EnD-SetSUMBT	None	64.09	3.29	6.45
	Total	68.25	3.36	6.45
EnD^2-SetSUMBT	None	66.25	3.35	6.25
	Knowledge	**71.61**	**3.52**	6.31

Table 3: Performance of the systems evaluated with real users. We have 550 dialogues for each system with success rates, ratings and the number of turns averaged.

(significant at $p < 0.05$). This suggests that in human interaction there is more data uncertainty, necessitating the knowledge uncertainty to be an explicit part of the model.

It is important to note here that solely a lower average number of turns is not necessarily an indicator of the desired behaviour of a system. For example, a system which says goodbye too early may never be successful, but will have a low average number of turns.

5 Conclusion

Whilst neural dialogue state trackers may achieve state-of-the-art performance in the isolated dialogue state tracking task, the absence of uncertainty estimates may lead to less robust performance of the downstream dialogue policy. In this work we propose the use of total and knowledge uncertainties along with confidence scores to form a dialogue belief state. We moreover describe a model, SetSUMBT, that can produce such a belief state via distillation. Experiments with both simulated and real users confirm that these uncertainty metrics can lead to more robust dialogue policy models. In future, we will investigate modifying span-based dialogue state trackers to incorporate uncertainty. We will further investigate the expansion of the SetSUMBT model to include the correlation between different domain-slot pairings.

Acknowledgements

CVN, MH, NL and SF are supported by funding provided by the Alexander von Humboldt Foundation in the framework of the Sofja Kovalevskaja Award endowed by the Federal Ministry of Education and Research. CG and HCL are supported by funds from the European Research Council (ERC) provided under the Horizon 2020 research and innovation programme (Grant agreement No. STG2018 804636). Google Cloud and HHU ZIM provided computational infrastructure.

References

Arsenii Ashukha, Alexander Lyzhov, Dmitry Molchanov, and Dmitry Vetrov. 2020. Pitfalls of in-domain uncertainty estimation and ensembling in deep learning. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Paweł Budzianowski, Tsung-Hsien Wen, Bo-Hsiang Tseng, Iñigo Casanueva, Ultes Stefan, Ramadan Osman, and Milica Gašić. 2018. Multiwoz - A Large-Scale Multi-Domain Wizard-of-Oz Dataset for Task-Oriented Dialogue Modelling. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics.

Guan-Lin Chao and Ian Lane. 2019. BERT-DST: Scalable end-to-end dialogue state tracking with bidirectional encoder representations from transformer. In *Proceedings of Interspeech 2019*, pages 1468–1472.

Zhi Chen, Lu Chen, Xiaoyuan Liu, and Kai Yu. 2020. Distributed structured actor-critic reinforcement learning for universal dialogue management. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 28:2400–2411.

Hyunjin Choi, Judong Kim, Seongho Joe, and Youngjune Gwon. 2021. Evaluation of BERT and ALBERT sentence embedding performance on downstream NLP tasks. In *Proceedings of the 25th International Conference on Pattern Recognition (ICPR)*, pages 5482–5487.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. *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)*, pages 4171–4186.

Mihail Eric, Rahul Goel, Shachi Paul, Abhishek Sethi, Sanchit Agarwal, Shuyang Gao, Adarsh Kumar, Anuj Goyal, Peter Ku, and Dilek Hakkani-Tur. 2020. MultiWOZ 2.1: A consolidated multi-domain dialogue dataset with state corrections and state tracking baselines. In *Proceedings of The 12th Language Resources and Evaluation Conference*, pages 422–428.

Yarin Gal. 2016. *Uncertainty in Deep Learning*. Ph.D. thesis, University of Cambridge.

Yarin Gal and Zoubin Ghahramani. 2016. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *Proceedings of the 33rd International Conference on Machine Learning*, volume 3, pages 1651–1660.

Milica Gašić, Simon Keizer, Francois Mairesse, Jost Schatzmann, Blaise Thomson, Kai Yu, and Steve Young. 2008. Training and evaluation of the HIS POMDP dialogue system in noise. SIGdial '08, page 112–119, USA. Association for Computational Linguistics.

Milica Gašić and Steve Young. 2014. Gaussian processes for POMDP-based dialogue manager optimization. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 22(1):28–40.

Michael Heck, Carel van Niekerk, Nurul Lubis, Christian Geishauser, Hsien-Chin Lin, Marco Moresi, and Milica Gašić. 2020. Trippy: A triple copy strategy for value independent neural dialog state tracking. In *Proceedings of the 21th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 35–44, 1st virtual meeting. Association for Computational Linguistics.

Matthew Henderson, Blaise Thomson, and Steve Young. 2013. Deep neural network approach for the dialog state tracking challenge. In *Proceedings of the SIGDIAL 2013 Conference*, pages 467–471, Metz, France. Association for Computational Linguistics.

Matthew Henderson, Blaise Thomson, and Steve Young. 2014. Robust dialog state tracking using delexicalised recurrent neural networks and unsupervised adaptation. In *Proceedings of the 2014 IEEE Spoken Language Technology Workshop (SLT)*, pages 360–365. IEEE.

Matthew S Henderson. 2015. *Discriminative methods for statistical spoken dialogue systems*. Ph.D. thesis, University of Cambridge.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*.

Jiaying Hu, Yan Yang, Chencai Chen, Liang He, and Zhou Yu. 2020. SAS: Dialogue state tracking via slot attention and slot information sharing. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6366–6375.

Filip Jurčíček, Blaise Thomson, and Steve Young. 2011. Natural actor and belief critic: Reinforcement algorithm for learning parameters of dialogue systems modelled as pomdps. *ACM Transactions on Speech and Language Processing (TSLP)*, 7(3):1–26.

Hwaran Lee, Seokhwan Jo, HyungJun Kim, Sangkeun Jung, and Tae-Yoon Kim. 2020. SUMBT+LaRL: End-to-end neural task-oriented dialog system with reinforcement learning. *arXiv preprint arXiv:2009.10447*.

Hwaran Lee, Jinsik Lee, and Tae-Yoon Kim. 2019. SUMBT: slot-utterance matching for universal and scalable belief tracking. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5478–5483.

Kyusong Lee, Tiancheng Zhao, Alan W. Black, and Maxine Eskenazi. 2018. Dialcrowd: A toolkit for easy dialog system assessment. In *Proceedings of the 19th Annual SIGdial Meeting on Discourse and Dialogue*, pages 245–248, Melbourne, Australia. Association for Computational Linguistics.

Esther Levin, Roberto Pieraccini, and Wieland Eckert. 1998. Using markov decision process for learning dialogue strategies. In *Proceedings of the 1998 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP'98 (Cat. No. 98CH36181)*, volume 1, pages 201–204. IEEE.

Shiyang Li, Semih Yavuz, Kazuma Hashimoto, Jia Li, Tong Niu, Nazneen Rajani, Xifeng Yan, Yingbo Zhou, and Caiming Xiong. 2020a. CoCo: Controllable counterfactuals for evaluating dialogue state trackers. *arXiv preprint arXiv:2010.12850*.

Ziming Li, Sungjin Lee, Baolin Peng, Jinchao Li, Julia Kiseleva, Maarten de Rijke, Shahin Shayandeh, and Jianfeng Gao. 2020b. Guided dialogue policy learning without adversarial learning in the loop. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2308–2317. Association for Computational Linguistics.

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

Andrey Malinin. 2019. *Uncertainty Estimation in Deep Learning with application to Spoken Language Assessment*. Ph.D. thesis, University of Cambridge.

Andrey Malinin and Mark Gales. 2018. Predictive uncertainty estimation via prior networks. In *Advances in Neural Information Processing Systems 31 (NeurIPS 2018)*, pages 7047–7058.

Andrey Malinin, Bruno Mlodozeniec, and Mark Gales. 2019. Ensemble distribution distillation. In *International Conference on Learning Representations*.

Nikola Mrkšić, Diarmuid Ó Séaghdha, Tsung-Hsien Wen, Blaise Thomson, and Steve Young. 2017a. Neural belief tracker: Data-driven dialogue state tracking. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1777–1788, Vancouver, Canada. Association for Computational Linguistics.

Nikola Mrkšić, Diarmuid Ó Séaghdha, Tsung-Hsien Wen, Blaise Thomson, and Steve Young. 2017b. Neural belief tracker: Data-driven dialogue state tracking. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1777–1788.

Yaniv Ovadia, Emily Fertig, Jie Ren, Zachary Nado, David Sculley, Sebastian Nowozin, Joshua V Dillon, Balaji Lakshminarayanan, and Jasper Snoek. 2019. Can you trust your model's uncertainty? evaluating predictive uncertainty under dataset shift. *Advances in Neural Information Processing Systems 32 (NeurIPS 2019)*.

Geoff Pleiss, Manish Raghavan, Felix Wu, Jon Kleinberg, and Kilian Q Weinberger. 2017. On fairness and calibration. *Advances in Neural Information Processing Systems*, 30:5680–5689.

Nina Poerner, Ulli Waltinger, and Hinrich Schütze. 2020. Sentence meta-embeddings for unsupervised semantic textual similarity. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7027–7034, Online. Association for Computational Linguistics.

Osman Ramadan, Paweł Budzianowski, and Milica Gašić. 2018. Large-scale multi-domain belief tracking with knowledge sharing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 432–437.

Max Ryabinin, Mark Gales, and Andrey Malinin. 2021. Scaling ensemble distribution distillation to many classes with proxy targets. *arXiv preprint arXiv:2105.06987*.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347.

Ryuichi Takanobu, Hanlin Zhu, and Minlie Huang. 2019. Guided dialog policy learning: Reward estimation for multi-domain task-oriented dialog. 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)*, pages 100–110.

Blaise Thomson, F Jurčíček, M Gašić, Simon Keizer, François Mairesse, Kai Yu, and Steve Young. 2010. Parameter learning for POMDP spoken dialogue models. In *Proceedings of the 2010 IEEE Spoken Language Technology Workshop*, pages 271–276.

Blaise Thomson and Steve Young. 2010. Bayesian update of dialogue state: A POMDP framework for spoken dialogue systems. *Computer Speech & Language*, 24(4):562–588.

Carel van Niekerk, Michael Heck, Christian Geishauser, Hsien-chin Lin, Nurul Lubis, Marco Moresi, and Milica Gasic. 2020. Knowing what you know: Calibrating dialogue belief state distributions via ensembles. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 3096–3102, Online. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Daan Wierstra, et al. 2016. Matching networks for one shot learning. *Advances in neural information processing systems*, 29:3630–3638.

Gellért Weisz, Paweł Budzianowski, Pei-Hao Su, and Milica Gašić. 2018. Sample efficient deep reinforcement learning for dialogue systems with large action spaces. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 26(11):2083–2097.

Jason Williams. 2012. A belief tracking challenge task for spoken dialog systems. In *NAACL-HLT Workshop on Future directions and needs in the Spoken Dialog Community: Tools and Data (SDCTD 2012)*, pages 23–24, Montréal, Canada. Association for Computational Linguistics.

Jason D Williams and Steve Young. 2007. Partially observable markov decision processes for spoken dialog systems. *Computer Speech & Language*, 21(2):393–422.

Fanghua Ye, Jarana Manotumruksa, Qiang Zhang, Shenghui Li, and Emine Yilmaz. 2021. Slot self-attentive dialogue state tracking. In *Proceedings of the Web Conference 2021*, pages 1598–1608.

Steve Young, Catherine Breslin, Milica Gašić, Matthew Henderson, Dongho Kim, Martin Szummer, Blaise Thomson, Pirros Tsiakoulis, and Eli Tzirkel Hancock. 2016. Evaluation of statistical POMDP-based dialogue systems in noisy environments. In *Situated Dialog in Speech-Based Human-Computer Interaction*, pages 3–14. Springer.

Steve Young, Jost Schatzmann, Karl Weilhammer, and Hui Ye. 2007. The hidden information state approach to dialog management. In *2007 IEEE International Conference on Acoustics, Speech and Signal Processing-ICASSP'07*, volume 4, pages IV–149. IEEE.

Jianguo Zhang, Kazuma Hashimoto, Chien-Sheng Wu, Yao Wang, S Yu Philip, Richard Socher, and Caiming Xiong. 2020. Find or classify? dual strategy for slot-value predictions on multi-domain dialog state tracking. In *Proceedings of the Ninth Joint Conference on Lexical and Computational Semantics*, pages 154–167.

Qi Zhu, Zheng Zhang, Yan Fang, Xiang Li, Ryuichi Takanobu, Jinchao Li, Baolin Peng, Jianfeng Gao, Xiaoyan Zhu, and Minlie Huang. 2020. Convlab-2: An open-source toolkit for building, evaluating, and diagnosing dialogue systems. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 142–149, Online. Association for Computational Linguistics.

A Example dialogues

Here we provide two example dialogues from our human trial, which illustrates the advantage of using uncertainty. In Figures 3 and 4 we observe that the system makes a second request to the user to clarify the intent of the user.

User: I need to book a cheap 4 star hotel.
System: Do you have a **area** preference?
User: I need the hotel to be **eastern**.
System: Is there a specific **area** of town you are interested in?
User: The **east**.

Figure 3: Extract from a dialogue with a real user illustrating the importance of uncertainty in a dialogue.

User: Find me a hotel in the city centre.
System: Do you have a price range preference?
User: **High**
System: It is in the centre. **What price range would you like?**
User: A **highly priced** hotel.
System: Everyone seems to enjoy Acorn guest house. **It is an expensive hotel**. Is there anything else I can help you with?

Figure 4: Extract from a dialogue with a real user illustrating the importance of uncertainty in a dialogue.

B Loss Functions

Consider the classification problem with input features $\boldsymbol{x}$, oracle class y and training dataset:

$$\mathcal{D}_{\text{train}} = \{(\boldsymbol{x}^{(1)}, y^{(1)}), ..., (\boldsymbol{x}^{(N)}, y^{(N)})\},$$

consisting of N labelled examples.

B.1 Label Smoothing Loss Function

The label smoothing loss is a regularised variant of standard negative log likelihood loss. Here, instead of considering a 1-hot target $\boldsymbol{y}_I$, some noise is induced in the target distribution in the form of:

$$\boldsymbol{y}_{ls} = (1 - \epsilon)\boldsymbol{y}_I + \frac{\epsilon}{K},$$

where ϵ is the smoothing parameter, $\boldsymbol{y}_{ls}$ the noisy/smoothed targets and $\boldsymbol{y}_I$ the one hot representation of the target y. The objective is to minimise the KL divergence between the predictive distribution, $\mathtt{P}(y|\boldsymbol{x}^{(i)}, \boldsymbol{\phi})$, and the smoothed target $\boldsymbol{y}_{ls}$. That is

$$\mathcal{L}_{ls}(\boldsymbol{\phi}, \mathcal{D}_{\text{train}}) = \frac{1}{N}\sum_{i=1}^{N} \mathtt{KL}\big[\boldsymbol{y}_{ls}^{(i)} || \mathtt{P}(y|\boldsymbol{x}^{(i)}, \boldsymbol{\phi})\big]$$

B.2 Distillation Loss Functions

Here we detail the loss functions used for ensemble distillation (EnD) and ensemble distribution distillation (EnD^2) in this work.

Consider an ensemble $\{\boldsymbol{\theta}^{(1)}, ..., \boldsymbol{\theta}^{(M)}\}$ consisting of M models, with predictive posterior $\mathtt{P}(y|\boldsymbol{x}^{(i)}, \mathcal{D}_{\text{train}})$.

Standard ensemble distillation (Hinton et al., 2015) is accomplished by minimising the KL-divergence between a student model with parameters $\boldsymbol{\phi}$ and the ensemble's predictive posterior:

$$\mathcal{L}_{\text{EnD}}(\boldsymbol{\phi}, \mathcal{D}_{\text{train}}) = \frac{1}{N}\sum_{i=1}^{N} \mathtt{KL}\big[\mathtt{P}(y|\boldsymbol{x}^{(i)}, \mathcal{D}_{\text{train}}) \,||\, \mathtt{P}(y|\boldsymbol{x}^{(i)}, \boldsymbol{\phi})\big]$$

Distribution distillation is accomplished using the improved loss function proposed by Ryabinin et al. (2021). Here, we first compute a *Proxy Dirichlet Target* with Dirichlet concentration parameters $\boldsymbol{\beta}$ from the ensemble:

$$\begin{aligned}
\hat{\pi}_k(\boldsymbol{x}) &= \frac{1}{M}\sum_{m=1}^{M} \mathtt{P}(y = \omega_k|\boldsymbol{x}, \boldsymbol{\theta}^{(m)}) \\
\tilde{\beta}_0(\boldsymbol{x}) &= \frac{K-1}{2\sum_{k=1}^{K}\hat{\pi}_k(\ln\hat{\pi}_k - \sum_{m=1}^{M}\frac{\ln\pi_k^{(m)}}{M})} \\
\beta_k(\boldsymbol{x}) &= \hat{\pi}_k(\boldsymbol{x})\cdot\tilde{\beta}_0(\boldsymbol{x}) + 1,\ \beta_0 = \sum_{k=1}^{K}\beta_k.
\end{aligned} \tag{12}$$

Given this *Proxy Dirichlet Target*, distribution distillation is done by minimising the following loss:

$$\begin{aligned}
&\mathcal{L}_{\text{EnD}^2}(\boldsymbol{\phi}, \mathcal{D}_{\text{train}}) = \\
&\frac{1}{N}\sum_{i=1}^{N}\Big[-\mathbb{E}_{\mathtt{p}(\boldsymbol{\pi}|\boldsymbol{x}^{(i)},\boldsymbol{\phi})}[\sum_{k=1}^{K}\hat{\pi}_k^{(i)}\ln\pi_k] \\
&+ \frac{1}{\beta_0^{(i)}}\mathtt{KL}[\mathtt{p}(\boldsymbol{\pi}|\boldsymbol{x}^{(i)}, \boldsymbol{\phi})\|\mathtt{p}(\boldsymbol{\pi}|\mathbf{1})]\Big].
\end{aligned} \tag{13}$$

C SetSUMBT Implementation Details

Here we provide details regarding the SetSUMBT model configuration and the model training configuration. Table 4 provides details about the configuration of the SetSUMBT model. Tables 5 and 6 provide details regarding the training configurations for both the single model and distillation (EnD) of SetSUMBT. For all SetSUMBT models the *Set Pooler* consists of a single convolutional layer with padding followed by a mean pooling layer.

Parameter	Value
Roberta pretrained checkpoint	`roberta-base`
Hidden size	768
SUM attention heads	12
Context tracking GRU hidden size	300
Set Pooler CNN filter size	3
Dropout rate	0.3
Maximum turn length	64
Candidate description length	12

Table 4: SetSUMBT model configuration.

Parameter	Value
Learning rate (LR)	$1e-5$
LR Scheduler warmup proportion	0.1
Batch size	3
Maximum turns per dialogue	12
Epochs	100
Early stopping criteria	25 epochs
Label smoothing ϵ	0.05
α_{goal}	1.0
α_{general}	0.2
α_{request}	0.2
α_{domain}	0.2

Table 5: Single model and EnD^2 training configurations. EnD^2 does utilise the Label smoothing ϵ.

D Variations in User Simulator Output

The user simulator used in our experiments consists of a natural language understanding (NLU) module, a rule based user agent and template based natural language generation module, all provided in the ConvLab 2 environment (Zhu et al., 2020). A pre-defined set of rules simulates the user behaviour based on the predicted semantic system actions and the resulting user actions are mapped to natural language using a pre-defined set of templates. To induce variation to the user simulator utterances and thus make understanding more difficult for the system, we utilise a set of pre-defined value variations obtained from the MultiWOZ 2.1 value map (Heck et al., 2020). For example, we can map the value, `expensive`, in the user action:

```
Inform - Restaurant - Price_range
        - expensive
```

to any of the following options:

```
[high end, high class, high
scale, high price, high priced,
 higher price, fancy, upscale,
  nice, expensively, luxury ].
```

In our experiments 20% of simulated user actions contain such variations.

Parameter	Value
Learning rate (LR)	$1e-5$
LR Scheduler warmup proportion	0.1
Batch size	3
Maximum turns per dialogue	12
Epochs	100
Early stopping criteria	25 epochs
Distribution smoothing	$1e-4$
Temperature scaling base temperature	2.5
Temperature scaling annealing cycle	0.1
α_{goal}	1.0
α_{general}	0.2
α_{request}	0.2
α_{domain}	0.2

Table 6: EnD training configuration.

E System Latencies

In this section we provide the processing times per turn for our SetSUMBT model as well as the systems used in this work. These processing times are averaged across the 7372 turns in the MultiWOZ 2.1 test set. This test is performed on a Google Cloud virtual machine containing a Nvidia V100 16GB GPU, 8 n1-standard VCPU's and 30GB memory. In Table 7 we compare the latencies of a single instance of SetSUMBT against a 10 model ensemble. In Table 8 we compare the latencies of the full dialogue system setups used in this work.

Tracker	Latency (ms)
Single instance	77.7680
10 Instance ensemble	768.0256

Table 7: Turn level latency of the SetSUMBT model and ensemble.

System	Latency (ms)
No uncertainty	135.9768
Confidence scores	138.0960
Total uncertainty	147.4574
Knowledge uncertainty	152.5392

Table 8: Turn level latency of the various full dialogue systems utilised in this work.

Chapter 7

CAMELL: Confidence-based Acquisition Model for Efficient Self-supervised Active Learning with Label Validation

This chapter summarises our work on uncertainty estimation in dialogue belief tracking and the downstream impact on the dialogue policy module and gives a verbatim copy of our manuscript (van Niekerk et al., 2023):

Carel van Niekerk et al. (2023). "CAMELL: Confidence-based Acquisition Model for Efficient Self-supervised Active Learning with Label Validation". In: *arXiv preprint arXiv:2310.08944 Version 1*. URL: https://arxiv.org/abs/2310.08944

7.1 Summary

In this work, we introduce CAMELL (**C**onfidence-based **A**cquisition **M**odel for **E**fficient self-supervised active **L**earning with **L**abel validation), a framework designed to mitigate these challenges and enhance both data and computational efficiency.

The CAMELL framework offers three major innovations: (1) **selective annotation**: It reduces the burden of data labelling by requiring expert annotators to label only a subset of the entire sequence, as opposed to the conventional approach of labelling every time-step and output category. (2) **self-supervision**: For the remaining parts of the sequence, the model employs a self-supervised approach to generate labels, thereby reducing dependency on human annotators. (3) **label validation**: Crucially, CAMELL incorporates a label validation mechanism that screens both expert and self-generated labels to ensure data quality. This prevents errors and inconsistencies from polluting the training set and deteriorating model performance.

We rigorously evaluated CAMELL across different tasks, with a special focus on dialogue belief trackinga task particularly sensitive to the quality and quantity of labeled data. Remarkably, our results indicate that CAMELL achieves 95% of full-training dataset performance while utilizing only 16% of expert-provided labels. In terms of efficiency and robustness, CAMELL significantly outperforms the existing baselines. Moreover, we noted that CAMELL exceeds the performance of existing active learning baselines in the context of machine translation with a generative language model. This underscores its potential efficacy for applications involving large language models. As an additional contribution, we developed a method for automatically detecting and correcting inaccuracies in human-provided labels, which further improves the overall quality of the dataset.

7.2 Personal Contributions

The implementation and technical results are my own work, while my co-authors contributed to the writing and proofreading process.

CAMELL: Confidence-based Acquisition Model for Efficient Self-supervised Active Learning with Label Validation

Carel van Niekerk, Christian Geishauser, Michael Heck, Shutong Feng
Hsien-chin Lin, Nurul Lubis, Benjamin Ruppik and Renato Vukovic and Milica Gašić
Heinrich Heine Universität Düsseldorf, Düsseldorf, Germany
{niekerk,geishaus,heckmi,fengs,linh,lubis,ruppik,revuk100,gasic}@hhu.de

Abstract

Supervised neural approaches are hindered by their dependence on large, meticulously annotated datasets, a requirement that is particularly cumbersome for sequential tasks. The quality of annotations tends to deteriorate with the transition from expert-based to crowd-sourced labelling. To address these challenges, we present **CAMELL** (**C**onfidence-based **A**cquisition **M**odel for **E**fficient self-supervised active **L**earning with **L**abel validation), a pool-based active learning framework tailored for sequential multi-output problems. CAMELL possesses three core features: (1) it requires expert annotators to label only a fraction of a chosen sequence, (2) it facilitates self-supervision for the remainder of the sequence, and (3) it employs a label validation mechanism to prevent erroneous labels from contaminating the dataset and harming model performance. We evaluate CAMELL on sequential tasks, with a special emphasis on dialogue belief tracking, a task plagued by the constraints of limited and noisy datasets. Our experiments demonstrate that CAMELL outperforms the baselines in terms of efficiency. Furthermore, the data corrections suggested by our method contribute to an overall improvement in the quality of the resulting datasets.

1 Introduction

Supervised deep neural networks require large amounts of accurately annotated data (Russakovsky et al., 2015; Szegedy et al., 2017; Li et al., 2020b). Sequential multi-output tasks, in particular, demand that the neural network makes multiple predictions at each time-step within an input sequence. Given this requirement, the effort to label data grows rapidly, exceeding what is practically feasible, as each individual time-step and every output category necessitates precise and consistent labelling. Therefore, the reliance on purely human-generated labels establishes a substantial constraint on the scalability of such systems.

A prominent example of a sequential multi-output label task for which this bottleneck is evident is dialogue belief tracking. A dialogue belief tracker is one of the core components of a dialogue system, tasked with inferring the goal of the user at every turn (Young et al., 2007). Current state-of-the-art trackers are based on deep neural network models (Lin et al., 2021; van Niekerk et al., 2021; Heck et al., 2022). These models outperform traditional Bayesian network-based belief trackers (Young et al., 2010; Thomson and Young, 2010). However, it is evident that these neural belief trackers are greatly hindered by the lack of adequate training data. Real-world conversations, even those pertaining to a specific task-oriented domain, are extremely diverse. They encompass a broad spectrum of user objectives, natural language variations, and the overall dynamic nature of human conversation. While there are many sources for dialogue data, such as call centres or virtual personal assistants, labelled dialogue data is scarce and several orders of magnitude smaller than, say, data for speech recognition (Panayotov et al., 2015) or translation (Bojar et al., 2017). Although zero-shot trackers do not require large amounts of labelled data, they typically underperform compared to supervised models that are trained on accurately labelled datasets (Heck et al., 2023).

One of the largest available labelled datasets for task-oriented dialogues is MultiWOZ, which is a multi-domain dialogue dataset annotated via crowd-sourcing. The challenges in achieving consistent and precise human annotations are apparent in all versions of MultiWOZ (Budzianowski et al., 2018; Eric et al., 2020; Zang et al., 2020; Han et al., 2021; Ye et al., 2022). Despite manual corrections in the most recent edition, model performance has

plateaued, not due to limitations in the models, but as a result of data inconsistencies (Li et al., 2020a).

In this work we present CAMELL, a pool-based active learning approach for sequential multi-output tasks. Given an underlying supervised learning model that can estimate confidence in its predictions, CAMELL substantially reduces the required labelling effort and is robust to annotation errors. CAMELL comprises of:

- A selection component that selects a subset of time-steps and output categories in input sequences to be labelled by experts rather than whole sequences, as is normally the case.
- A self-supervision component that uses self-generated labels for the remaining time-steps and output categories within selected input sequences.
- A label validation component which examines the reliability of the human-provided labels.

We assess the feasibility of CAMELL's selection and self-supervision components within an idealised setting for machine translation. Encouraged by the results of the feasibility study, we then apply all components of CAMELL to the dialogue belief tracking task. Notably, we achieve $\mathbf{95}\%$ of a tracker's full-training dataset performance using merely $\mathbf{16}\%$ of the expert-provided labels.

On top of this framework, we develop a method for automatically detecting and correcting inaccuracies of human labels in existing datasets. We illustrate that these corrections boost performance of two distinct tracking models, overcoming the limitations imposed by labelling inconsistencies. Having demonstrated its efficacy in machine translation and dialogue belief tracking, our framework holds potential for broad applicability across various sequential multi-output tasks, such as object tracking, pose detection, and language modelling.

2 Related Work

2.1 Active Learning

Active learning is a machine learning framework that identifies under-represented scenarios in labelled data and interactively queries an annotator (Cohn et al., 1996). This framework employs an acquisition function to select the most beneficial data points for querying. This function estimates the potential improvement in performance resulting from an observed label. These functions typically depend on prediction uncertainty (Houlsby et al., 2011), data space coverage (Sener and Savarese, 2018), variance reduction (Johansson et al., 2007), or topic popularity (Iovine et al., 2022).

Active learning approaches can be categorised into stream-based and pool-based (Settles, 2009). Stream-based setups are usually employed when data creation and labelling occur simultaneously. In contrast, pool-based approaches separate these steps, operating under the assumption that an unlabelled data pool is available.

Active learning has been frequently employed in tasks such as image classification (Houlsby et al., 2011; Gal et al., 2017) and machine translation (Vashistha et al., 2022; Liu et al., 2018). A noteworthy example in machine translation is the work of Hu and Neubig (2021), which enhances efficiency by applying active learning to datasets enriched with frequently used phrases. While this approach reduces the requirement for exhaustive labelling, it constrains the annotator's capacity to produce comprehensive translations of longer sentences.

Conversely, active learning is less prevalent in dialogue belief tracking, with Xie et al. (2018) being a notable exception. Their framework involves querying labels for complete sequences (dialogues) and bases selection on a single output category, neglecting any potential correlation between categories. Furthermore, this approach does not address label verification, hence not accounting for annotation quality problems.

One work that addresses the problem of annotation quality within an active learning framework is Su et al. (2018). In this work, stream-based active learning is deployed for the purpose of learning whether a dialogue is successful. The user provided labels are validated using a label confidence score. This innovative learning strategy is however not directly applicable to sequential multi-output tasks as it does not deal with the sequential nature of the problem.

2.2 Semi-Supervised Learning

In a semi-supervised learning setting, a model learns from a mixture of labelled and unlabelled data. This is typically achieved by training an encoder-decoder type model using a large pool of unlabelled data. This "pre-trained" encoder model is then combined with a task-specific architecture to perform supervised learning during the so called

"fine-tuning" stage. The training of many belief tracking models is based on fine-tuning transformer models such as RoBERTa (Liu et al., 2019), being a prime example of semi-supervised learning (van Niekerk et al., 2021; Lee et al., 2021; Su et al., 2022; Heck et al., 2022).

Pseudo labelling (Lee, 2013) and noisy student model training (Xie et al., 2020) are alternative semi-supervised methodologies where an initial *teacher* model, trained using a small labelled dataset, produces pseudo labels for unlabelled data. This data is then used to train a *student* model, which eventually assumes the *teacher* role. This cyclical training procedure can bolster model performance without necessitating extra data.

2.3 Label Validation

The process of manually correcting labels is very tedious and expensive. As a result, many works focus on learning from imperfect labels, using loss functions and/or model architectures adapted for label noise (Reed et al., 2015; Xiao et al., 2015; Sukhbaatar et al., 2015). Still these methods have been unable to match the performance of models trained on datasets that include manually corrected labels. However, the alternative of automated label validation or correction is often overlooked by such works. It has been shown that learning from automatically corrected labels, e.g. based on confidence scores, perform better than learning from noisy labels alone (Liu et al., 2017; Jiao et al., 2019). The major drawback of these approaches is that they rely on overconfident predictions of neural network models to correct labels, which can further bias the model.

3 CAMELL: Confidence-based Acquisition Model for Efficient Self-supervised Active Learning with Label Validation

In this section, we introduce our pool-based active learning approach, named *CAMELL*, to address sequential multi-output classification problems. Let us consider a classification problem with input features, denoted as $\boldsymbol{x}$, and output $\boldsymbol{y}$. According to Read et al. (2015), this problem can be termed as a multi-output classification problem if the output consists of multiple label categories that need to be predicted simultaneously. Specifically, for a problem with M categories, the output is represented as $\boldsymbol{y} = \langle y^1, y^2, ..., y^M \rangle$, where each $y^m, m \in [1, M]$ can be binary or multivariate. Furthermore, this problem is characterised as a sequential classification problem if the output is dependent on a sequence of prior inputs. For a sequence with T time-steps, the input-output pairs can be represented as $\langle \{\boldsymbol{x}_1, \boldsymbol{y}_1\}, \{\boldsymbol{x}_2, \boldsymbol{y}_2\}, \cdots, \{\boldsymbol{x}_T, \boldsymbol{y}_T\} \rangle$, where $\boldsymbol{y}_t = \langle y_t^1, y_t^2, ..., y_t^M \rangle$ represents the output labels at time step $t \in [1, T]$.

In a conventional setting, for an unlabelled data sequence $\boldsymbol{X}_i = \langle \boldsymbol{x}_1, \cdots, \boldsymbol{x}_T \rangle$, an annotator would typically be required to provide labels, y_t^m, for each label category $m \in [1, M]$ at every time step $t \in [1, T]$.

3.1 Requirements

CAMELL, as a confidence-based active learning framework, utilises confidence estimates to determine the data points to be queried for labelling. Hence, a critical requirement for the learning model within CAMELL is its ability to estimate the confidence associated with its predictions. More specifically, for each category m and each value $v \in \mathcal{V}^m$ this category can assume, the learning model should be able to estimate the predictive probability (an indicator of model confidence), $\pi_t^m(v) = \mathsf{p}\,(y_t^m = v)$, as part of the predictive distribution, $\boldsymbol{\pi}_t^m = [\pi_t^m(v)]_{\forall v \in \mathcal{V}^m}$.

The calibration of these confidence estimates is also critical. Calibration refers to the alignment between the model's estimated confidence and the empirical likelihood of its predictions (Desai and Durrett, 2020). Should the model's confidence estimates be poorly calibrated, it may select instances that are not informative, resulting in an inefficient allocation of the annotation budget and potentially sub-optimal performance.

3.2 Active Learning Approach

The approach we propose starts with an initial learning model, which is trained using a small labelled *seed* dataset and iteratively progresses through four stages: data selection, labelling, label validation, and semi-supervised learning. These iterations continue until either a pre-defined performance threshold is achieved or the dataset is fully labelled. The schematic representation of this approach is illustrated in Figure 1.

Stage 1: Data selection In each cycle, we select a subset of N_{sel} sequences from the unlabelled pool of size N_{unlb}. Selection is based on the model's prediction confidence, p_t^m. Instances in which the

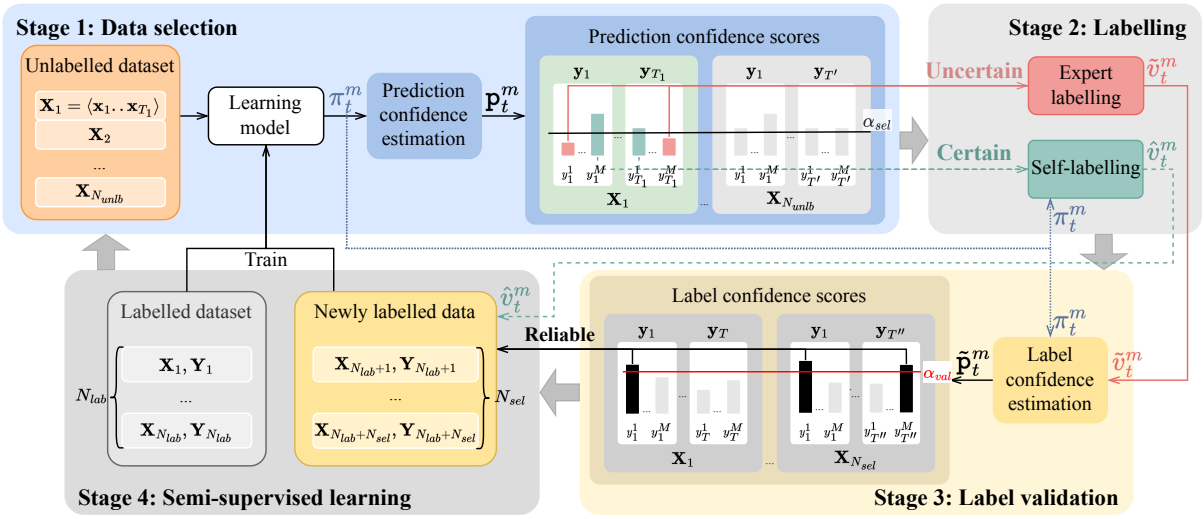


Figure 1: CAMELL comprises four stages. Stage 1 involves data selection, choosing instances for labelling where the model displays uncertainty (confidence below the α_{sel} threshold), as indicated by pink arrows. In Stage 2, annotators label the selected instances while the model self-labels the remaining ones (dashed green arrows). Stage 3 validates labels using a label confidence estimate, incorporating only those exceeding the α_{val} threshold and the self-labelled data into the dataset (black arrows). Finally, Stage 4 involves retraining the model for the next cycle.

model displays low confidence (confidence below the α_{sel} threshold) are selected. More precisely, an input sequence is selected if the model shows high uncertainty for at least one time-step t and label category m instance y_t^m. The α_{sel} threshold is set such that N_{sel} sequences are selected for labelling.

Stage 2: Labelling Expert annotators are responsible for labelling the time-steps and categories selected in Stage 1. These labels are denoted by $\tilde{v}_t^m$. Concurrently, the model completes the labelling for the remaining time-steps and categories, $\hat{v}_t^m$, where its confidence is always above the threshold α_{sel}.

Stage 3: Label validation We can consider the expert labels, $\tilde{v}_t^m$, with label confidence, $\tilde{\mathrm{p}}_t^m$, below a threshold α_{val} to be potentially incorrect. To safeguard the model from being trained with these potentially erroneous labels, we purposely exclude them (these labels are masked in the dataset). The α_{val} threshold can be set using a development set, which helps maintain the quality of the training data.

Stage 4: Semi-supervised learning At each iteration of the active learning approach the expert provided labels that passed validation (Stage 3) and the self-determined labels from Stage 2 are added to the labelled pool, resulting in $N_{\text{lab}} + N_{\text{sel}}$ data sequences. Based on these the learning model is retrained.

3.3 Confidence Estimation

In order to accurately estimate the prediction confidence used in stage 1 of CAMELL and the label confidence used in stage 3, we propose a confidence estimation model for each. In order for these models to accurately capture the confidence of a learning model, we incorporate both the *total* and *knowledge* uncertainty of the learning model. *Total* uncertainty captures all uncertainty in the model's prediction, irrespective of the source. Conversely, *knowledge* uncertainty in a model originates from its incomplete understanding, which occurs due to a lack of relevant data during training, or the inherent complexity of the problem (Gal, 2016).

Both the prediction and label confidence estimation models share the same objective: to estimate the probability that the value v_t^m for a specific label category m at time-step t is correct. To provide the training data for these models, we assume that the labels in the labelled pool are correct, as they have already been validated. Furthermore, we retrain these models whenever more labelled data is obtained.

Broadly speaking, both models share the same structure:

$$\begin{aligned} \boldsymbol{h}_t^m &= \mathsf{Enc}_{\texttt{Intra-Cat}}(\boldsymbol{z}_t^m) \\ \boldsymbol{h}_t &= \mathsf{Enc}_{\texttt{Inter-Cat}}([\boldsymbol{z}_t^j]_{j=1}^M) \\ \mathrm{p}_t^m &= \mathsf{Conf}(\boldsymbol{h}_t^m, \boldsymbol{h}_t), \end{aligned} \tag{1}$$

where $\boldsymbol{z}_t^m = [\pi_t^m(v_t^m), \mathcal{T}(\boldsymbol{\pi}_t^m), \mathcal{K}(\boldsymbol{\pi}_t^m)]$ is a set of uncertainty measures for category m. As illustrated in Figure 2, these measures consist of the predictive probability specific to $\pi_t^m(v_t^m)$, along with measures of *total*, $\mathcal{T}(\boldsymbol{\pi}_t^m)$, and *knowledge* uncertainty, $\mathcal{K}(\boldsymbol{\pi}_t^m)$, associated with the predictive distribution $\boldsymbol{\pi}_t^m$. The *intra-category encoder* is tasked with extracting important category specific features, $\boldsymbol{h}_t^m$, from these uncertainties. Important features across categories, $\boldsymbol{h}_t$, are extracted by the *inter-category encoder*[1]. The *inter-category* encoder allows the model to take advantage of any correlations between categories, which was not done by Xie et al. (2018). Both the *inter-* and *intra-category* encoders are linear fully connected layers. The *confidence estimation* component generates a confidence score, p_t^m, reflecting the accuracy of a given category's value. This component is composed of a linear feature transformation layer, followed by a prediction layer with a Sigmoid activation function.

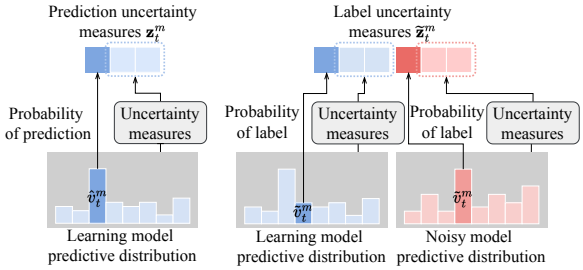
The design choices for the confidence estimation models were motivated by a desire to capture both *intra-* and *inter-category* uncertainty for reliable confidence estimation. We observed that excluding *inter-category* features degraded performance, emphasising the importance of incorporating them.

3.3.1 Prediction Confidence Estimation

The objective of the prediction confidence estimation model is to assess whether the value predicted, $\hat{v}_t^m = \arg\max_{v \in \mathcal{V}^m}(\pi_t^m(v))$, by the learning model is the "true" value, based on the prediction confidence score p_t^m. This model, also known as the confidence-based acquisition model, is used as the criterion for selecting instances to label in stage 1.

3.3.2 Label Confidence Estimation

The objective of the label confidence estimation model is to determine whether an annotator's label, $\tilde{v}_t^m$, is the "true" value, with this decision being based on the label confidence score $\tilde{\mathrm{p}}_t^m$. Although it is possible to use a single model for both prediction and label confidence estimation[2], but we believe it is a sub-optimal strategy. This strategy is sub-optimal because the model has not been exposed to instances of "incorrect" labels. To address this, we generate a *noisy* dataset featuring "incorrect" labels for training purposes.



(a) Prediction (b) Label

Figure 2: Category-specific uncertainty measures: (a) displays prediction uncertainty, including prediction probability and total and knowledge uncertainty; (b) depicts label uncertainty, including label probability and total and knowledge uncertainty from both learning and noisy models.

Further, we extend $\tilde{\boldsymbol{z}}_t^m$ to include uncertainty measures drawn from both a *noisy* model, trained on the corresponding *noisy* dataset, and the original learning model (as depicted in Figure 2b). Given that the *noisy* model is conditioned to accept the "incorrect" labels as correct, the discrepancy in uncertainty between the *noisy* model and the learning model enhances the label confidence estimator's ability to identify potentially incorrect labels.

Noisy dataset The creation of a noisy dataset can be approached in two ways. One method is to randomly replace a portion of labels. However, this approach may not yield a realistic noisy dataset, considering human errors are rarely random. A second approach, particularly when the learning model is an ensemble, as is often the case for uncertainty-endowed deep learning models (Gal and Ghahramani, 2016; Ashukha et al., 2020), is to leverage individual ensemble members to supply noisy labels. This method may be more effective, given the individual members' typical lower accuracy compared to the ensemble as a whole.

In our proposed approach, we initially select α_{noise} percent of the sequences from the training data at random. For each category m, we choose a random ensemble member to generate noisy labels. This ensemble member creates labels at each time step t by sampling from its predictive probability distribution. To avoid generating labels from the *clean* dataset, the probabilities of these are set to zero prior to sampling. The noisy dataset is regenerated after each update of the learning model

[1] During label confidence estimation, for categories not selected for labelling, self-labels are used to complete the *inter-category* features.

[2] For example, in Su et al. (2018) the confidence score of the learning model is directly used for both purposes.

using the updated ensemble members, enhancing diversity of noisy labels.

4 Experiments

4.1 Baselines

Random selection randomly selects sequences to be annotated. Random selection is often used as a baseline for active learning approaches as it allows us to observe the impact of purely adding more labelled data to our labelled pool without strategically selecting sequences to be labelled. Its advantage is that it maintains the full data distribution with every selection, thus not creating a bias (Dasgupta and Hsu, 2008).

Bayesian Active Learning by Disagreement (BALD) is an uncertainty-based active learning method which employs knowledge uncertainty as the primary metric for selection (Houlsby et al., 2011). This technique has established itself as a strong baseline in various applications. For instance, in image classification tasks (Gal et al., 2017) and named entity recognition (Shen et al., 2017), BALD has shown notable performance. Its performance is further enhanced when used in conjunction with ensemble models (Beluch et al., 2018). Given its widespread adoption and proven efficacy, we see BALD as an ideal baseline, as it is one of the most effective methods for active learning currently available.

In our study, we incorporate two variants of BALD. The first, referred to as the baseline **BALD**, involves selecting complete sequences for labelling. We examined two criteria for making the selection decision: one based on the cumulative uncertainty across all time-steps and label categories, and another based on the average uncertainty across categories and time. Upon evaluation, we observed that the latter criterion yielded superior results, and therefore, adopted it as our baseline.

We further present an enhanced version of BALD which consists of stages $1, 2$, and 4 of our approach as outlined in Section 3, utilising knowledge uncertainty as the *prediction confidence estimate*. We call this BALD with self supervision, **BALD+SS**. However, knowledge uncertainty is not suitable for label validation as it cannot provide candidate level confidence scores.

4.2 Feasibility Study

We conduct a feasibility study in an idealised setting by applying our proposed **CAMEL** (CAMELL without label validation) framework to the machine translation task. Neural machine translation (NMT) is fundamentally aimed at translating sentences from a source language to a target language.

4.2.1 Implementation Details

We apply CAMEL to the machine translation task using the T5 encoder-decoder transformer model (`t5-small`) (Raffel et al., 2020). We utilise an ensemble of 10 models in order to produce a well-calibrated predictive distribution, which requires 2500 GPU hours to fully train. Approximately 40% of this time is utilised to train the ensemble, 50% for the annotation process described below, and 10% for training the confidence estimator. The ensemble estimates two types of uncertainty. The first is the total uncertainty, represented by the entropy of the predictive distribution. The second is knowledge uncertainty, which refers to the mutual information between the predictive distribution and the ensemble members. The WMT17 DE-EN dataset, which consists of German to English translations (Bojar et al., 2017), is used for training, and METEOR (Banerjee and Lavie, 2005) serves as the evaluation metric.

As machine translation does not entail a multi-output task, we employed a simplified version of the confidence estimation model, introduced in Section 3.3, consisting of only the intra-category encoder. The latent dimension of the encoder and feature transformation layer is 16. The parameters are optimised using the standard binary negative log likelihood loss (Cox, 1958)[3].

It is crucial to address the inherent challenges in sequential machine translation labelling: (1) future sentence structure and labels can change depending on the current label, and (2) for any sentence position there exists multiple valid candidate words. This complexity hinders seamless integration of dataset labels with the learning model for new data labelling. To avoid high translation annotation costs, we propose a practical approach: using an *expert* translation model, specifically the MBART-50 multilingual model (Tang et al., 2020), to simulate a human annotator.

The labelling process we propose, depicted in Figure 3, is a multi-stage procedure. Initially, the learning model produces a translation for a selected source language sentence. As it generates the translation, it simultaneously estimates its confidence

[3] `https://gitlab.cs.uni-duesseldorf.de/general/dsml/camell.git`

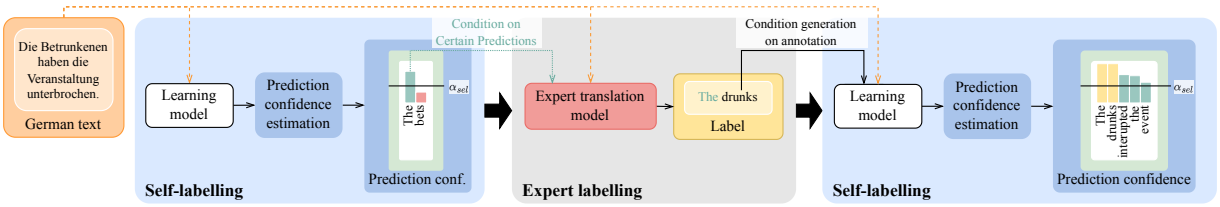
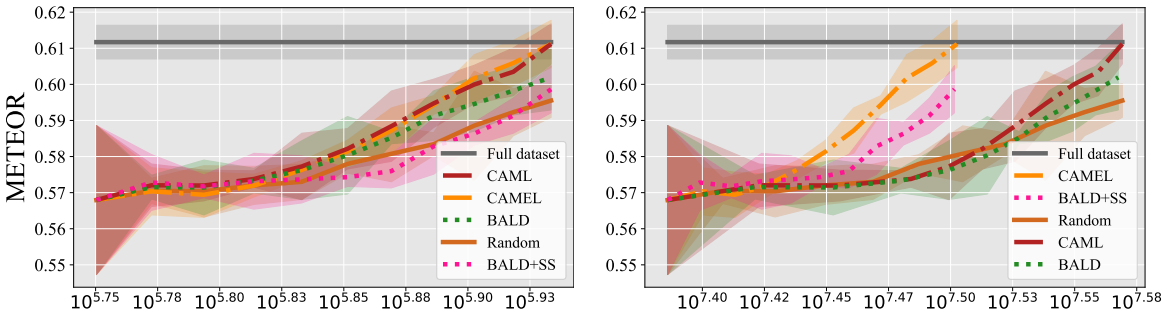


Figure 3: The model-based annotation process for semi-supervised annotation. The learning model initiates the translation with the word "The", then confidence for the next token generation is below the threshold. The expert annotation model is prompted and provides the next word, "drunks". The learning model resumes and successfully generates the remainder of the translation: "interrupted the event".



(a) Number of translations

(b) Number of labels

Figure 4: METEOR score of the T5 translation model using different active learning approaches on the WMT17 DE-EN test set, as a function of (a) the number of translations and (b) the number of labels, with 95% conf. int.

for the subsequent token. Should this confidence fall below a set threshold α_{sel}, the *expert* translation model steps in to supply the next word in the translation. After the label is provided, the learning model resumes the translation generation. For any future token whose confidence drops below the threshold, the *expert* translation model re-engages. This process continues until a complete translation for the source sentence is realised. The uncertainty threshold, α_{sel}, is strategically chosen to yield a maximum of N_{ann} word labels.

4.2.2 Results

We establish that our proposed CAMEL framework, enhanced with self-supervision, is significantly more efficient in label acquisition than baseline models like BALD, BALD+SS, and random selection. This efficiency is evident in Figure 4b, which showcases CAMEL's need for fewer labels to achieve similar performance levels.

A notable point to observe in Figure 4a is that the introduction of self-supervision to CAMEL does not significantly influence its performance in terms of the number of translations, as evidenced by the comparison between CAML (CAMEL without the self-supervised labelling component) and CAMEL. This implies that self-supervision within CAMEL is applied predominantly when the model's predictions are considered reliable. In contrast, we observe that BALD+SS, despite its label efficiency shown in Figure 4b, performs poorly in terms of translations, as demonstrated in Figure 4a. This drop in performance may be attributed to BALD+SS's tendency to incorrectly self-label complex examples. This trend is further validated by CAML's lower expected calibration error (ECE) in Table 1.

In Figure 4 we observe that, CAMEL and its variant CAML exceed the performance of their BALD counterparts. This emphasises the value of our proposed confidence-based acquisition model in selecting instances for labelling.

Regardless of the methodology used, all models require roughly the same number of translations, as shown in Figure 4a. This supports the widely accepted notion that exposure to large datasets is vital for training robust natural language processing (NLP) models.

Encouraged by these results, we adapt CAMEL to address the dialogue belief tracking problem, which struggles with significantly noisier datasets.

Confidence Estimator	Dataset	ECE (%) ↓
CE-SetSUMBT + CAML	MultiWOZ 2.1	**9.65∗**
CE-SetSUMBT + BALD	MultiWOZ 2.1	17.21
CE-T5 + CAML	WMT17 DE-EN	**26.74∗**
CE-T5 + BALD	WMT17 DE-EN	47.21

Table 1: Comparison of the expected calibration error (ECE) of confidence estimation approaches. $*$ indicates significant difference on 95% conf. int.

4.3 Dialogue Belief Tracking Task

In task-oriented dialogue, the dialogue ontology $\mathcal{O}$ contains a set of M `domain-slot` pairs $\{s^1, s^2, ..., s^M\}$ and a set of plausible values $\mathcal{V}_{s^m}$ for each s^m. The goal of the dialogue belief tracker is to infer the user's intention for each s_m by predicting a probability distribution over the plausible values. Notably, each set of plausible values, $\mathcal{V}_{s^m}$, includes the `not_mentioned` value, indicating that a specific `domain-slot` pair is not part of the user's goal. This allows for computing the model's confidence for slots not present in the goal.

To train a belief tracking model, we require the dialogue state, which includes the `value` for each `domain-slot`, in every dialogue turn. The dialogue state at turn t in dialogue i is represented as $\mathcal{B}_{t,i} = \{(s^m, v_{t,i}^{s^m})\}_{s^m \in \mathcal{O}}$, where $v_{t,i}^{s^m}$ denotes the `value` for the `domain-slot` pair s^m at turn t in dialogue i. Consequently, we obtain a dataset $\mathcal{D} = \{(\mathbf{u}_{i,1:t}^{\text{usr}}, \mathbf{u}_{i,1:t-1}^{\text{sys}}, \mathcal{B}_{t,i})_{t=1}^{T_i}\}_{i=1}^{D}$, consisting of D dialogues, each comprising T_i turns, where user and system utterances at turn t in dialogue i are denoted as $\mathbf{u}_{i,t}^{\text{usr}}$ and $\mathbf{u}_{i,t}^{\text{sys}}$, respectively.

To create dataset $\mathcal{D}$, annotators usually provide relevant values for the `domain-slot` pairs they believe are present in the user's utterance at every turn t. Subsequently, a handcrafted rule-based tracker considers the previous state $\mathcal{B}_{t-1}$, the semantic actions present in the system utterance and the values provided by the annotator to generate complete dialogue states for each turn (Budzianowski et al., 2018). However, this approach has several drawbacks. Firstly, rule-based trackers tend to be imprecise and necessitate redevelopment for each new application, making it less versatile. Secondly, it may not use the time of human annotators efficiently as the learning model could potentially predict the state for a substantial part of the dialogue accurately. Lastly, there is the risk of human annotators inadvertently overlooking some slots in the user input, which could result in missing or incomplete data.

4.3.1 Learning Model

To apply *CAMELL* (CAMEL with Label validation) to the dialogue belief tracking problem, we use the CE-SetSUMBT (Calibrated Ensemble - SetSUMBT) model (van Niekerk et al., 2021). We use CE-SetSUMBT because it produces well-calibrated uncertainty estimates, which is important for CAMELL. The CE-SetSUMBT model consists of 10 ensemble members, requiring 1000 GPU hours to fully train. Approximately 45% of this time is utilised for training the ensemble, 45% for training the *noisy* model, and 10% for training the confidence estimators.

The CE-SetSUMBT model employs an ensemble of models to generate uncertainty estimates. By utilising marginal distributions, it provides well-calibrated probability distributions of values that each `domain-slot` pair s^m can take: $\mathsf{p}(v_t^{s^m} = v|\mathbf{u}_t^{\text{usr}}, \mathbf{u}_{t-1}^{\text{sys}}, \boldsymbol{c}_{t-1}^{s^m})$, where $\boldsymbol{c}_{t-1}^{s^m}$ represents the dialogue context. It also estimates the total and knowledge uncertainty present in the belief state distribution.

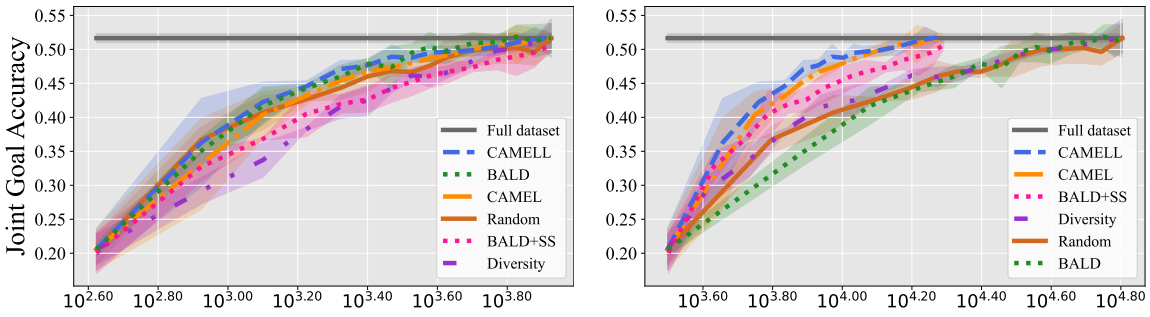
4.3.2 Datasets

In order to test our proposed approach we utilise the multi-domain task-oriented dialogue dataset MultiWOZ 2.1 (Eric et al., 2020; Budzianowski et al., 2018) and its manually corrected test set provided in version 2.4 (Ye et al., 2022). Therefore, in our experiments, we regard MultiWOZ 2.1 as a dataset with substantial label noise, and the test set of MultiWOZ 2.4 a dataset with accurate labels.

4.3.3 Implementation Details

The latent dimension of the intra- and inter-category encoders and feature transformation layer is 16. During training of the label confidence estimation model (Section 3.3.2), to avoid overfitting, we deploy techniques to improve the calibration of this model. These include binary label smoothing loss (Szegedy et al., 2016), temperature scaling and noisy training using Gaussian noise (An, 1996).

For the *seed* dataset (Section 3) we randomly select 5% of dialogues on which we train the initial CE-SetSUMBT model. The other dialogues in the dataset are treated as the unlabelled pool. At each update step another 5% of the data are selected to be labelled. At each point where we require expert labels, we take the original labels provided in the dataset.



(a) Number of dialogues

(b) Number of labels

Figure 5: Joint goal accuracy of the CE-SetSUMBT model using different active learning approaches, on the MultiWOZ 2.1 test set, as a function of (a) the number of dialogues and (b) the number of labels, with 95% conf. int.

4.3.4 Evaluation

As the main metric for our experiments we use joint goal accuracy (JGA) (Henderson et al., 2014). We further include the joint goal expected calibration error (ECE) (Guo et al., 2017; van Niekerk et al., 2020), which measures the calibration of the model. In terms of measuring efficiency of each method, we examine JGA as a function of the number of expert provided labels. In order to assess the quality of the corrected dataset, we measure the JGA of the methods trained on a noisy dataset, with and without the proposed label correction.

4.3.5 Dialogue Diversity Baseline

We include an additional dialogue diversity baseline, aiming to obtain labels for dialogues geometrically dissimilar from those in the labelled pool, thus ensuring data space coverage. This diversity strategy proposed by Xie et al. (2018) assesses similarity based on vector embeddings of the candidate dialogue versus labelled dialogues. We adapt this approach by employing RoBERTa model embeddings (Liu et al., 2019), fine-tuned unsupervisedly on the MultiWOZ dialogues for masked language modeling tasks.

4.3.6 Results

As shown in Figure 5b, our proposed CAMELL framework, along with CAMEL, requires significantly fewer labels to reach performance levels comparable to those of the baseline methods. This indicates that CAMELL is more efficient in dialogue belief tracking than the baseline strategies, a finding that aligns with our observations in translation tasks. It also highlights the vital role played by CAMELL's confidence estimates in guiding the active learning process. This conclusion is further supported by the lower calibration error of CAMELL's confidence estimates, as demonstrated in Table 1.

Another key observation, evident in Figure 5, is that CAMELL requires fewer dialogues and labels than CAMEL to achieve comparable performance. This showcases the effectiveness of label validation in CAMELL, allowing the model to discard labels that may otherwise hinder its learning.

It is important to reiterate, paralleling our findings in machine translation, that regardless of the framework, all models necessitate approximately the same number of dialogues for training, as depicted in Figure 5a.

4.3.7 Label Correction

Using CAMELL's label validation scheme (Section 3.3.2), we automatically correct labels in the MultiWOZ 2.1 training set. This process involves: (1) training a CE-SetSUMBT belief tracker on MultiWOZ 2.1, (2) identifying potentially incorrect labels through label validation, and (3) replacing these labels with predictions of CE-SetSUMBT (labels are only replaced if the prediction confidence is higher than label confidence). The quality of the cleaned dataset is evaluated using two distinct trackers.

We retrain the CE-SetSUMBT belief tracker from scratch on this cleaned dataset (CE-SetSUMBT+LC) for the first evaluation. In Table 2, we present the JGA of the two CE-SetSUMBT trackers on two test sets: the (noisy) MultiWOZ 2.1 test-set and the (manually corrected) MultiWOZ 2.4 test-set[4]. On the MultiWOZ 2.1 test-set, the model trained using label correction shows

[4]We never use the MultiWOZ 2.4 validation-set during training.

Model Setup	MultiWOZ 2.1		MultiWOZ 2.4	
	JGA (%) ↑	ECE (%) ↓	JGA (%) ↑	ECE (%) ↓
CE-SetSUMBT	51.79	10.09	61.63	7.27
CE-SetSUMBT + LC	52.83	11.58	**63.32∗**	6.33
TripPy	55.28	–	64.45	–
TripPy + LC	56.11	–	**66.02∗**	–

Table 2: Comparison of joint goal accuracy (JGA) and expected calibration error (ECE), of trackers trained with and without label corrections (LC). ∗ indicates significant difference on 95% conf. int.

marginal improvements, which is not surprising as the MultiWOZ 2.1 test-set contains errors, and hence cannot be used to adequately assess the effect of label correction. In contrast, on the MultiWOZ 2.4 test-set, the model trained using label correction shows statistically significant improvements. To further substantiate the effectiveness of our label correction, we train the TripPy state tracker (Heck et al., 2020) using the MultiWOZ 2.1 training dataset with and without the label corrections. We again observe a significant improvement in performance on the MultiWOZ 2.4 test-set. This demonstrates that the dataset resulting from label correction is of significantly higher quality.

In our investigation of the improved dataset, we identified three prevalent label errors, which our approach successfully rectifies, as demonstrated in Table 3. (I) Hallucinated annotations, where the annotator assigns labels not present in the dialogue context, (II) Multi-annotation, the case of assigning multiple labels to the same piece of information, and (III) Erroneous annotation, the situation where an incorrect label is assigned based on the context. These instances underscore the efficacy of our label validation model in minimising the propagation of errors into the dataset.

5 Conclusion

We propose CAMELL, a novel active learning approach that incorporates elements of self-supervision and label validation, aimed at minimising the dependency on labelled data for tackling sequential multi-output labelling problems, while simultaneously filtering out untrustworthy labels. We have demonstrated the feasibility of our framework in an idealised setting, applying it to the machine translation task to showcase its potential. Through extensive experimentation in the more realistic setting for the dialogue belief tracking task, we have shown that our approach significantly outperforms baselines in robustness and data efficiency. As a spin-off, we present a methodology for automated

Error Type	Conversation	MultiWOZ 2.1 Labels and Corrections
I	*User:* I would like to find a place that serves **moderately** priced **Chinese** food.	{Restaurant: {Food: Chinese, (95%) Price: Moderate, (94%) Day: Tuesday, (11%) Day: not_mentioned}} (72%)
II	*User:* I feel like going to a **nightclub**. *System:* Okay, the **Soul Tree Nightclub** is a popular place. Would you like the address or phone number? *User:* I will appreciate that.	{Attraction: {Type: Night club, (94%) Name: Soul Tree}, (53%) Hotel: {Name: Sou, (14%) Name: not_mentioned}} (34%)
III	*User:* I need a train leaving on **Friday** and I want to get there by **21 : 30**. Leaving **Broxbourne**.	{Train: {Dept.: Broxbourne, (94%) Day: Friday, (95%) Arrive by: 21:20, (1%) Arrive by: 21:30}} (83%)

Table 3: Examples of three common types of annotation errors in the MultiWOZ 2.1 dataset detected and corrected by CAMELL, (I) hallucinated annotations, (II) multi-annotation and (III) erroneous annotation. For each we provide the confidence scores of the labels and the corrections proposed by the model. Incorrect labels are marked in red and the proposed corrections in blue.

dataset correction, and our experiments confirm that our label correction method enhances the overall quality of a dataset.

We believe that this work has far reaching implications. Firstly, it underscores the indispensable role of uncertainty estimation in learning models, advocating for the development of more efficient computational methods in deep learning. This is particularly important given the computational cost of ensemble models, which is a significant limitation in works relying on accurate uncertainty estimation. Secondly, the versatility of CAMELL opens up possibilities for its application across diverse sequential multi-output labelling problems. Thirdly, it demonstrates that, in principle, dataset deficiencies can be addressed via data-driven approaches, circumventing the need for extensive manual or rule-based curation. This is particularly pertinent considering the prevailing belief that undesirable outcomes produced by NLP models are inherently linked to the training datasets and cannot be rectified algorithmically (Eisenstein, 2019, 14.6.3).

Looking ahead, we anticipate that refining the process of generating *noisy* datasets could result in a model capable of not only identifying label noise but also filtering out biases, false premises, and misinformation. Further, investigating more efficient methods for uncertainty estimation will be pivotal in minimising the computational costs of CAMELL in large-scale applications.

Acknowledgements

C. van Niekerk, M. Heck, S. Feng and N. Lubis are supported by funding provided by the Alexander von Humboldt Foundation in the framework of the Sofja Kovalevskaja Award endowed by the Federal Ministry of Education and Research, while C. Geishauser, H-C. Lin, B. Ruppik and R. Vukovic are supported by funds from the European Research Council (ERC) provided under the Horizon 2020 research and innovation programme (Grant agreement No. STG2018804636). We thank Andrey Malinin for his valuable inputs in the work.

References

Guozhong An. 1996. The Effects of Adding Noise During Backpropagation Training on a Generalization Performance. *Neural Computation*, 8(3):643–674.

Arsenii Ashukha, Alexander Lyzhov, Dmitry Molchanov, and Dmitry Vetrov. 2020. Pitfalls of In-Domain Uncertainty Estimation and Ensembling in Deep Learning. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Satanjeev Banerjee and Alon Lavie. 2005. METEOR: An Automatic Metric for MT Evaluation with Improved Correlation with Human Judgments. In *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, Ann Arbor, Michigan. Association for Computational Linguistics.

W. H. Beluch, T. Genewein, A. Nurnberger, and J. M. Kohler. 2018. The Power of Ensembles for Active Learning in Image Classification. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Los Alamitos, CA, USA. IEEE Computer Society.

Ondřej Bojar, Rajen Chatterjee, Christian Federmann, Yvette Graham, Barry Haddow, Shujian Huang, Matthias Huck, Philipp Koehn, Qun Liu, Varvara Logacheva, Christof Monz, Matteo Negri, Matt Post, Raphael Rubino, Lucia Specia, and Marco Turchi. 2017. Findings of the 2017 Conference on Machine Translation (WMT17). In *Proceedings of the Second Conference on Machine Translation*. Association for Computational Linguistics.

Paweł Budzianowski, Tsung-Hsien Wen, Bo-Hsiang Tseng, Iñigo Casanueva, Stefan Ultes, Osman Ramadan, and Milica Gašić. 2018. MultiWOZ - A Large-Scale Multi-Domain Wizard-of-Oz Dataset for Task-Oriented Dialogue Modelling. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 5016–5026. Association for Computational Linguistics.

David A Cohn, Zoubin Ghahramani, and Michael I Jordan. 1996. Active Learning with Statistical Models. *Journal of Artificial Intelligence Research (JAIR)*, 4:129–145.

David R Cox. 1958. The Regression Analysis of Binary Sequences. *Journal of the Royal Statistical Society: Series B (Methodological)*, 20(2):215–232.

Sanjoy Dasgupta and Daniel Hsu. 2008. Hierarchical Sampling for Active Learning. In *Proceedings of the 25th International Conference on Machine Learning*, page 208–215. Association for Computing Machinery.

Shrey Desai and Greg Durrett. 2020. Calibration of Pre-trained Transformers. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 295–302.

Jacob Eisenstein. 2019. *Introduction to Natural Language Processing*. MIT Press.

Mihail Eric, Rahul Goel, Shachi Paul, Abhishek Sethi, Sanchit Agarwal, Shuyang Gao, Adarsh Kumar, Anuj Goyal, Peter Ku, and Dilek Hakkani-Tur. 2020. MultiWOZ 2.1: A Consolidated Multi-Domain Dialogue Dataset with State Corrections and State Tracking Baselines. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 422–428, Marseille, France. European Language Resources Association.

Yarin Gal. 2016. *Uncertainty in Deep Learning*. Ph.D. thesis, University of Cambridge.

Yarin Gal and Zoubin Ghahramani. 2016. Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning*, volume 3, pages 1651–1660.

Yarin Gal, Riashat Islam, and Zoubin Ghahramani. 2017. Deep Bayesian Active Learning with Image Data. In *International Conference on Machine Learning*, pages 1183–1192. PMLR.

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. 2017. On Calibration of Modern Neural Networks. In *Proceedings of the 34th International Conference on Machine Learning*, pages 1321–1330.

Ting Han, Ximing Liu, Ryuichi Takanabu, Yixin Lian, Chongxuan Huang, Dazhen Wan, Wei Peng, and Minlie Huang. 2021. MultiWOZ 2.3: A Multi-domain Task-Oriented Dialogue Dataset Enhanced with Annotation Corrections and Co-Reference Annotation. In *CCF International Conference on Natural Language Processing and Chinese Computing*, pages 206–218. Springer.

Michael Heck, Nurul Lubis, Benjamin Ruppik, Renato Vukovic, Shutong Feng, Christian Geishauser, Hsien-chin Lin, Carel van Niekerk, and Milica Gasic. 2023. ChatGPT for Zero-shot Dialogue State Tracking: A Solution or an Opportunity? In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 936–950, Toronto, Canada. Association for Computational Linguistics.

Michael Heck, Nurul Lubis, Carel van Niekerk, Shutong Feng, Christian Geishauser, Hsien-Chin Lin, and Milica Gašić. 2022. Robust Dialogue State Tracking with Weak Supervision and Sparse Data. *Transactions of the Association for Computational Linguistics*, 10:1175–1192.

Michael Heck, Carel van Niekerk, Nurul Lubis, Christian Geishauser, Hsien-Chin Lin, Marco Moresi, and Milica Gašić. 2020. TripPy: A Triple Copy Strategy for Value Independent Neural Dialog State Tracking. In *Proceedings of the 21th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 35–44. Association for Computational Linguistics.

Matthew Henderson, Blaise Thomson, and Jason D. Williams. 2014. The Second Dialog State Tracking Challenge. In *Proceedings of the 15th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*, pages 263–272, Philadelphia, PA, U.S.A. Association for Computational Linguistics.

Neil Houlsby, Ferenc Huszar, Zoubin Ghahramani, and Máté Lengyel. 2011. Bayesian Active Learning for Classification and Preference Learning. *arXiv preprint arXiv:1112.5745 Version 1*.

Junjie Hu and Graham Neubig. 2021. Phrase-level Active Learning for Neural Machine Translation. In *Proceedings of the Sixth Conference on Machine Translation*, pages 1087–1099, Online. Association for Computational Linguistics.

Andrea Iovine, Pasquale Lops, Fedelucio Narducci, Marco de Gemmis, and Giovanni Semeraro. 2022. An empirical evaluation of active learning strategies for profile elicitation in a conversational recommender system. *Journal of Intelligent Information Systems*, 58(2):337–362.

Yang Jiao, Shahram Latifi, and Mei Yang. 2019. Self Error Detection and Correction for Noisy Labels Based on Error Correcting Output Code in Convolutional Neural Networks. In *2019 IEEE 9th Annual Computing and Communication Workshop and Conference (CCWC)*, pages 0311–0316.

Ulf Johansson, Tuve Lofstrom, and Lars Niklasson. 2007. The Importance of Diversity in Neural Network Ensembles - An Empirical Investigation. In *2007 International Joint Conference on Neural Networks*, pages 661–666.

Chia-Hsuan Lee, Hao Cheng, and Mari Ostendorf. 2021. Dialogue State Tracking with a Language Model using Schema-Driven Prompting. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 4937–4949, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Dong-Hyun Lee. 2013. Pseudo-Label : The Simple and Efficient Semi-Supervised Learning Method for Deep Neural Networks. In *International Conference on Machine Learning (ICML) 2013 Workshop : Challenges in Representation Learning (WREPL)*.

Shiyang Li, Semih Yavuz, Kazuma Hashimoto, Jia Li, Tong Niu, Nazneen Rajani, Xifeng

Yan, Yingbo Zhou, and Caiming Xiong. 2020a. CoCo: Controllable Counterfactuals for Evaluating Dialogue State Trackers. In *International Conference on Learning Representations (ICLR)*.

Xiaoya Li, Jingrong Feng, Yuxian Meng, Qinghong Han, Fei Wu, and Jiwei Li. 2020b. A Unified MRC Framework for Named Entity Recognition. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5849–5859. Association for Computational Linguistics.

Zhaojiang Lin, Bing Liu, Seungwhan Moon, Paul Crook, Zhenpeng Zhou, Zhiguang Wang, Zhou Yu, Andrea Madotto, Eunjoon Cho, and Rajen Subba. 2021. Leveraging Slot Descriptions for Zero-Shot Cross-Domain Dialogue State Tracking. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics.

Ming Liu, Wray Buntine, and Gholamreza Haffari. 2018. Learning to actively learn neural machine translation. In *Proceedings of the 22nd Conference on Computational Natural Language Learning*, pages 334–344, Brussels, Belgium. Association for Computational Linguistics.

Xin Liu, Shaoxin Li, Meina Kan, Shiguang Shan, and Xilin Chen. 2017. Self-Error-Correcting Convolutional Neural Network for Learning with Noisy Labels. In *2017 12th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2017)*, pages 111–117.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A Robustly Optimized BERT Pretraining Approach. *arXiv preprint arXiv:1907.11692 Version 1*.

Carel van Niekerk, Michael Heck, Christian Geishauser, Hsien-chin Lin, Nurul Lubis, Marco Moresi, and Milica Gašić. 2020. Knowing What You Know: Calibrating Dialogue Belief State Distributions via Ensembles. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 3096–3102, Online. Association for Computational Linguistics.

Carel van Niekerk, Andrey Malinin, Christian Geishauser, Michael Heck, Hsien-chin Lin, Nurul Lubis, Shutong Feng, and Milica Gašić. 2021. Uncertainty Measures in Neural Belief Tracking and the Effects on Dialogue Policy Performance. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Vassil Panayotov, Guoguo Chen, Daniel Povey, and Sanjeev Khudanpur. 2015. Librispeech: An ASR corpus based on public domain audio books. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5206–5210.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *The Journal of Machine Learning Research*, 21(1).

Jesse Read, Luca Martino, Pablo M. Olmos, and David Luengo. 2015. Scalable multi-output label prediction: From classifier chains to classifier trellises. *Pattern Recognition*, 48(6):2096–2109.

Scott E Reed, Honglak Lee, Dragomir Anguelov, Christian Szegedy, Dumitru Erhan, and Andrew Rabinovich. 2015. Training Deep Neural Networks on Noisy Labels with Bootstrapping.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. 2015. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision*, 115(3):211–252.

Ozan Sener and Silvio Savarese. 2018. Active Learning for Convolutional Neural Networks: A Core-Set Approach. In *International Conference on Learning Representations*.

Burr Settles. 2009. Active Learning Literature Survey. Technical report, University of Wisconsin-Madison, Department of Computer Sciences.

Yanyao Shen, Hyokun Yun, Zachary Lipton, Yakov Kronrod, and Animashree Anandkumar. 2017.

Deep Active Learning for Named Entity Recognition. In *Proceedings of the 2nd Workshop on Representation Learning for NLP*, Vancouver, Canada. Association for Computational Linguistics.

Pei-Hao Su, Milica Gašić, and Steve Young. 2018. Reward estimation for dialogue policy optimisation. *Computer Speech and Language*, 51:24–43.

Yixuan Su, Lei Shu, Elman Mansimov, Arshit Gupta, Deng Cai, Yi-An Lai, and Yi Zhang. 2022. Multi-Task Pre-Training for Plug-and-Play Task-Oriented Dialogue System. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4661–4676, Dublin, Ireland. Association for Computational Linguistics.

Sainbayar Sukhbaatar, Joan Bruna, Manohar Paluri, Lubomir Bourdev, and Rob Fergus. 2015. Training Convolutional Networks with Noisy Labels. In *3rd International Conference on Learning Representations, ICLR 2015 (Workshop)*.

Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander A. Alemi. 2017. Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, AAAI'17, page 4278–4284. AAAI Press.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. 2016. Rethinking the Inception Architecture for Computer Vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2818–2826.

Yuqing Tang, Chau Tran, Xian Li, Peng-Jen Chen, Naman Goyal, Vishrav Chaudhary, Jiatao Gu, and Angela Fan. 2020. Multilingual Translation with Extensible Multilingual Pretraining and Finetuning. *arXiv preprint arXiv:2008.00401 Version 1*.

Blaise Thomson and Steve Young. 2010. Bayesian update of dialogue state: A POMDP framework for spoken dialogue systems. *Computer Speech and Language*, 24(4):562–588.

Neeraj Vashistha, Kriti Singh, and Ramakant Shakya. 2022. Active Learning for Neural Machine Translation. *arXiv preprint arXiv:2301.00688 Version 1*.

Tong Xiao, Tian Xia, Yi Yang, Chang Huang, and Xiaogang Wang. 2015. Learning from Massive Noisy Labeled Data for Image Classification. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2691–2699.

Kaige Xie, Cheng Chang, Liliang Ren, Lu Chen, and Kai Yu. 2018. Cost-Sensitive Active Learning for Dialogue State Tracking. In *Proceedings of the 19th Annual SIGdial Meeting on Discourse and Dialogue*, pages 209–213, Melbourne, Australia. Association for Computational Linguistics.

Qizhe Xie, Minh-Thang Luong, Eduard Hovy, and Quoc V. Le. 2020. Self-Training With Noisy Student Improves ImageNet Classification. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10684–10695.

Fanghua Ye, Jarana Manotumruksa, and Emine Yilmaz. 2022. MultiWOZ 2.4: A Multi-Domain Task-Oriented Dialogue Dataset with Essential Annotation Corrections to Improve State Tracking Evaluation. In *Proceedings of the 23rd Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 351–360, Edinburgh, UK. Association for Computational Linguistics.

Steve Young, Milica Gašić, Simon Keizer, François Mairesse, Jost Schatzmann, Blaise Thomson, and Kai Yu. 2010. The Hidden Information State model: A practical framework for POMDP-based spoken dialogue management. *Computer Speech & Language*, 24(2):150–174.

Steve Young, Jost Schatzmann, Karl Weilhammer, and Hui Ye. 2007. The Hidden Information State Approach to Dialog Management. In *2007 IEEE International Conference on Acoustics, Speech and Signal Processing - ICASSP '07*, volume 4, pages IV–149–IV–152.

Xiaoxue Zang, Abhinav Rastogi, Srinivas Sunkara, Raghav Gupta, Jianguo Zhang, and Jindong Chen. 2020. MultiWOZ 2.2 : A Dialogue Dataset with Additional Annotation Corrections

and State Tracking Baselines. In *Proceedings of the 2nd Workshop on Natural Language Processing for Conversational AI*, pages 109–117, Online. Association for Computational Linguistics.

Chapter 8

Conclusion

The objective of this thesis was to address the lack of uncertainty estimation in dialogue belief tracking, whilst also illustrating the importance of uncertainty in this task. In this final chapter, the aim is to summarise the contributions, discuss their implications, highlight some limitations of this work, and suggest avenues for future research.

8.1 Summary of Key Findings

In this thesis, three works are presented which address the issues of uncertainty estimation in dialogue belief tracking. By integrating uncertainty into the decision-making and the knowledge acquisition processes of our models, the role of uncertainty in developing more effective and robust dialogue systems is illustrated.

In Chapter 5 we enhanced a baseline dialogue belief tracker using the label-smoothing objective trained ensemble. This model achieves state-of-the-art calibration and exceptional top-3 prediction accuracy of 84.08%. Furthermore, our model showcases the best L2-norm performance, indicating that the quality of predicted uncertainty is as crucial as the average JGA. Notably, our calibration techniques are broadly applicable to any neural dialogue belief tracking method.

In Chapter 6 we tackled the absence of uncertainty estimates in state-of-the-art neural dialogue state trackers, which can hinder downstream dialogue policy performance. We introduced the concept of using total and knowledge uncertainties along with confidence scores to form a more comprehensive dialogue belief state. Our model, SetSUMBT, employs distillation techniques to effectively capture these elements of uncertainty. Experiments with both simulated and real users validated that incorporating these uncertainty metrics significantly improves the robustness of downstream dialogue policy models.

In Chapter 7 we introduce CAMELL, an active learning framework designed to minimise dependency on labelled data in sequential multi-output problems. CAMELL incorporates elements of both self-supervision and label validation, effectively filtering out untrustworthy annotations. This innovation was validated in two distinct applications: machine translation, where it showcased its potential, and more crucially, in dialogue belief tracking, where it significantly outperformed existing baselines in terms of both robustness and data efficiency.

Moreover, CAMELL offers an automated method for dataset correction, a notable spin-off that promises to improve the quality of labelled data, as confirmed by our experiments. This work carries three far-reaching implications. First, it elevates the role of uncertainty estimation in machine learning models, calling for more computational efficiency in deep learning, especially considering the often prohibitive costs of ensemble models. Second, the versatility of CAMELL presents a broad spectrum of applications across various sequential multi-output labeling problems. Third, and perhaps most importantly, our work challenges the prevailing notion that deficiencies in datasets are irreparable algorithmically, offering a data-driven alternative to labor-intensive manual or rule-based correction.

8.2 Limitations

We identified a couple of important limitations related to the work presented. By highlighting these limitations, we aim to provide a balanced perspective on our contributions and to guide future work in addressing these challenges. First, while our techniques have demonstrated effectiveness in certain dialogue state tracking architectures, they have yet to be adapted and tested for span-prediction based state trackers, a dominant paradigm in dialogue state tracking (DST).

Further, all three contributions, ranging from the calibration techniques in multi-domain dialogue belief trackers to the CAMELL framework, come with a significant computational overhead, primarily due to ensemble-based modelling. This computational burden poses challenges for scalability and real-world implementation, particularly for larger scale systems and more complex models.

8.3 Recommendations for Future Research

Despite advances in uncertainty estimation, such as Bayesian neural network, ensembles, etc., achieving accurate and efficient uncertainty estimation remains an elusive goal. We identify important directions for future work including efficient uncertainty estimation, enhancing the trustworthiness of Large Language Models (LLMs), and innovative applications of uncertainty estimates in LLMs' learning processes.

8.3.1 Uncertainty Estimation

Efficient Uncertainty Estimation

Existing state-of-the-art techniques for uncertainty estimation, particularly ensemble methods, are computationally expensive, especially when scaled to larger applications. Emerging approaches such as Neural Processes, MOPED Bayesian Neural Networks, and Epistemic Neural Networks could potentially bridge this gap, ensuring reliable uncertainty estimation without the computational cost of ensemble methods (Garnelo et al., 2018; Krishnan et al., 2020; Osband et al., 2023). An investigation into the adaptability and performance of such methods in complex NLP tasks could potentially transform uncertainty estimation in this domain.

Uncertainty Estimation in Dialogue State Tracking Models

In this thesis, we applied uncertainty estimation techniques to picklist-style dialogue belief tracking models. These models, however, are overshadowed by the superior performance of more advanced span-prediction and generation model-based dialogue state trackers (Heck et al., 2022; Lin et al., 2021b). Despite their proficiency, these advanced models have a significant limitation, their inability to generate uncertainty estimates.

Our findings emphasise the important role of reliable confidence estimates. They are instrumental not only for enhancing the performance of downstream modules within a dialogue system but also for improving the label-efficiency and adaptability of DST models. This revelation illuminates a crucial gap in DST models and calls for the incorporation of uncertainty estimation into span-prediction and generation model based DSTs.

One promising approach could involve the fusion of predictive distributions from various prediction mechanisms within span-prediction based DST models. When combined with strategic sampling methods, such models could be empowered to offer dependable confidence estimates (Shelmanov et al., 2021).

Regarding generation model-based DSTs, transforming token distributions into dialogue belief states presents a complex, unsolved challenge. Nevertheless, our experiments with the CAMELL active learning framework demonstrate that incorporating uncertainty estimation into such models is feasible. In light of these findings, extending the CAMELL framework to cater to generation models

or span-prediction based models for dialogue state tracking emerges as a promising avenue for future research.

8.3.2 Trustworthiness of Large Language Models

As Large Language Models (LLMs) like ChatGPT become increasingly woven into diverse facets of society, the quest for reliability and trustworthiness gains urgency. One major problem of such models is their tendency to generate incorrect or made up information, referred to as hallucination. Further, these models are unable to produce reliable confidence scores for the text they generate, and are often overconfident in such hallucinations (Xiong et al., 2023). This makes it difficult for users to know when they can trust model predictions. Improved uncertainty quantification in deep learning models can yield dual benefits: it enhances AI accountability and serves as a crucial tool for flagging generated text that may contain hallucinations or false premises.

The label validation technique of CAMELL (outlined in Chapter 7) utilises model uncertainty to validate user labels. While this technique is effective for current applications, it relies on the accurate measurement of uncertainties, a feature that current LLMs are yet unable to provide. If this hurdle is overcome, the mechanism could be adapted to automatically trigger regeneration of low-confidence responses, offering a potential avenue for identifying and correcting inaccuracies in text generated by LLMs.

8.3.3 Uncertainty Estimation in Large Language Models

The complexity and enormity of current Large Language Models (LLMs) like ChatGPT and LLama are both a blessing and a curse (Touvron et al., 2023). Boasting more than 50 billion parameters, these models are computationally expensive to use. Further these models generally, do not produce well calibrated confidence scores for the text they generate (Xiong et al., 2023). Due to their computational complexity, it is not practically possible to apply uncertainty estimation techniques which require ensembling, or even complete model retraining. Future research should focus on innovative and scalable methods for uncertainty estimation in such LLMs (Chen and Mueller, 2023).

Opportunities arising from reliable uncertainty estimates in LLMs are:

Active Learning: Utilising uncertainty estimates for active learning could optimise the selection process of examples for both fine-tuning and in-context learning (a powerful capability of LLMs where models can incorporate new knowledge without fine-tuning of their parameters (Brown et al., 2020)), ensuring that the models' evolution is not just rapid but also directed and meaningful. This could significantly reduce the training time needed to update such large models, lowering both the cost and, more importantly, the environmental footprint.

Rating Reliability: Integrating uncertainty estimates into Large Language Models (LLMs) can significantly enhance the evaluation of the reliability of both the content generated by these models and the feedback they receive from users. This integration is crucial because there have been instances where human feedback has actually impaired the performance of models like ChatGPT. This is often due to the integration of poor-quality feedback into the model's fine-tuning process (Casper et al., 2023; Chen et al., 2023). With access to uncertainty estimates, a label validation approach like the one proposed in CAMELL (see Chapter 7) could filter out harmful feedback and potentially prevent the deterioration of the model's abilities.

Moreover, when models are not corrected, they can "learn" from their own mistakes or "hallucinations," leading to the repeated generation of incorrect information. In contrast, receiving feedback from other LLMs has been shown to be an effective method for improving model performance (Lee et al., 2023). When coupled with reliable uncertainty estimates, this approach could become even more powerful, ensuring that the models are refined and enhanced with high-quality, reliable information, reducing the risk of amplifying errors or biases.

Model Collapse Prevention: Research indicates that Large Language Models (LLMs) can deviate from producing natural language when continually learning from their own outputs, a phenomenon known as model collapse (Shumailov et al., 2023). Data and knowledge uncertainties could be pivotal in taking on this challenge. Data uncertainty, stemming from the inherent ambiguities and variations in the source data, mirrors the natural diversity in language. Knowledge uncertainty, on the other hand, reflects the model's informational gaps.

A promising avenue for future exploration involves developing active learning and training approaches that maintain data uncertainty, preserving the richness and diversity of language, while reducing knowledge uncertainty. This balanced approach could bolster the model's informational depth and accuracy without compromising the natural fluidity and diversity of language expression.

In the short term, research efforts should concentrate on developing scalable, computationally-efficient uncertainty estimation methods for both classification and generative models. These methods must also be woven into accountable AI frameworks and be capable of real-time label validation to safeguard against misinformation and hallucinations. Long-term objectives should position uncertainty as an integral feature of LLMs and other large-scale neural networks. Such integrated uncertainty could not only bolster the trustworthiness of large-scale systems but also enhance their ability to interact with humans in a more natural manner.

Appendix A

Supplementary Proofs

A.1 Deep Learning

Theorem 1 (Linear Transformation of Positional Encodings). *Let $\boldsymbol{e}_t \in \mathbb{R}^d$ be a positional encoding vector defined as:*

$$\boldsymbol{e}_t = \begin{bmatrix} \sin(\lambda_0 t) & \cos(\lambda_0 t) & \sin(\lambda_1 t) & \cos(\lambda_1 t) & \cdots & \sin\left(\lambda_{\frac{d}{2}-1} t\right) & \cos\left(\lambda_{\frac{d}{2}-1} t\right) \end{bmatrix}$$

where $\lambda_m = 10000^{\frac{-2m}{d}}$ for $m = 0, 1, \ldots, \frac{d}{2} - 1$. Then there exists a linear transformation $\boldsymbol{T}^{(k)} \in \mathbb{R}^{d \times d}$:

$$\boldsymbol{T}^{(k)} = \begin{bmatrix} \Phi_0^{(k)} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \Phi_1^{(k)} & \cdots & \mathbf{0} \\ \cdots & \cdots & \cdots & \cdots \\ \mathbf{0} & \mathbf{0} & \cdots & \Phi_{\frac{d}{2}-1}^{(k)} \end{bmatrix},$$

$$\Phi_m^{(k)} = \begin{bmatrix} \cos(\lambda_m k) & \sin(\lambda_m k) \\ -\sin(\lambda_m k) & \cos(\lambda_m k), \end{bmatrix}$$

such that:

$$\boldsymbol{T}^{(k)} \boldsymbol{e}_t = \boldsymbol{e}_{t+k},$$

for all $k \in [1 - t, N - t]$ and $t \in [1, N]$ where N is the length of the sequence of observations.

Proof. We start from the trigonometric angle sum identities for sine and cosine:

$$\begin{aligned} \sin(\lambda_m(t+k)) &= \sin(\lambda_m t + \lambda_m k) \\ &= \sin(\lambda_m t)\cos(\lambda_m k) + \cos(\lambda_m t)\sin(\lambda_m k) \\ \cos(\lambda_m(t+k)) &= \cos(\lambda_m t + \lambda_m k) \\ &= \cos(\lambda_m t)\cos(\lambda_m k) - \sin(\lambda_m t)\sin(\lambda_m k). \end{aligned}$$

This implies that we can rewrite each element in the positional encoding $\boldsymbol{e}_{t+k}$ in terms of elements in $\boldsymbol{e}_t$ and fixed trigonometric constants as follows:

$$\begin{bmatrix} \sin(\lambda_m(t+k)) \\ \cos(\lambda_m(t+k)) \end{bmatrix} = \begin{bmatrix} \cos(\lambda_m k) & \sin(\lambda_m k) \\ -\sin(\lambda_m k) & \cos(\lambda_m k) \end{bmatrix} \begin{bmatrix} \sin(\lambda_m t) \\ \cos(\lambda_m t) \end{bmatrix}.$$

This defines a rotation matrix $\Phi_m^{(k)}$ to capture this linear relationship:

$$\Phi_m^{(k)} \begin{bmatrix} \sin(\lambda_m t) \\ \cos(\lambda_m t) \end{bmatrix} = \begin{bmatrix} \sin(\lambda_m(t+k)) \\ \cos(\lambda_m(t+k)). \end{bmatrix}.$$

For each frequency component $m = 0, 1, \ldots, \frac{d}{2} - 1$, we can then stack these individual rotation matrices $\Phi_m^{(k)}$ into a global transformation matrix $\boldsymbol{T}^{(k)}$:

$$\boldsymbol{T}^{(k)} = \begin{bmatrix} \Phi_0^{(k)} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \Phi_1^{(k)} & \cdots & \mathbf{0} \\ \cdots & \cdots & \cdots & \cdots \\ \mathbf{0} & \mathbf{0} & \cdots & \Phi_{\frac{d}{2}-1}^{(k)} \end{bmatrix}.$$

In conclusion, we have shown that a linear transformation matrix $\boldsymbol{T}^{(k)}$ exists, which satisfies $\boldsymbol{T}^{(k)}\boldsymbol{e}_t = \boldsymbol{e}_{t+k}$ for all $k \in [1-t, N-t]$ and $t \in [1, N]$ where N is the length of the sequence of observations.

□

Theorem 2 (Similarity between positional encodings)**.** *Consider a positional encoding vector $\boldsymbol{e}_t$ of dimension d, defined as:*

$$\boldsymbol{e}_t = \begin{bmatrix} \sin(\lambda_0 t) & \cos(\lambda_0 t) & \sin(\lambda_1 t) & \cos(\lambda_1 t) & \cdots & \sin\left(\lambda_{\frac{d}{2}-1} t\right) & \cos\left(\lambda_{\frac{d}{2}-1} t\right) \end{bmatrix},$$

where $\lambda_m = 10000^{\frac{-2m}{d}}$. The dot-product similarity between the positional encoding vectors $\boldsymbol{e}_t$ and $\boldsymbol{e}_{t+k}$ is given by:

$$\boldsymbol{e}_t \cdot \boldsymbol{e}_{t+k} = \sum_{m=0}^{\frac{d}{2}-1} \cos(\lambda_m k),$$

for all $k \in [1-t, N-t]$ and $t \in [1, N]$ where N is the length of the sequence of observations. Further $\boldsymbol{e}_t \cdot \boldsymbol{e}_{t+k} = \boldsymbol{e}_t \cdot \boldsymbol{e}_{t-k}$ for all $k \in [0, N-t]$ and $t \in [1, N]$.

Proof. Let us first consider a 2-dimensional positional encoding:

$$\boldsymbol{e}_t = \begin{bmatrix} \sin(\lambda_0 t) & \cos(\lambda_0 t) \end{bmatrix}.$$

We can use the angle sum equations for sine and cosine to find the dot-product similarity for the 2-dimensional case:

$$\begin{aligned} \boldsymbol{e}_t \cdot \boldsymbol{e}_{t+k} &= \sin(\lambda_0 t)\sin(\lambda_0(t+k)) + \cos(\lambda_0 t)\cos(\lambda_0(t+k)) \\ &= \sin(\lambda_0 t)\left[\sin(\lambda_0 t)\cos(\lambda_0 k) + \cos(\lambda_0 t)\sin(\lambda_0 k)\right] \\ &\quad + \cos(\lambda_0 t)\left[\cos(\lambda_0 t)\cos(\lambda_0 k) - \sin(\lambda_0 t)\sin(\lambda_0 k)\right] \\ &= \sin^2(\lambda_0 t)\cos(\lambda_0 k) + \sin(\lambda_0 t)\cos(\lambda_0 t)\sin(\lambda_0 k) \\ &\quad + \cos^2(\lambda_0 t)\cos(\lambda_0 k) - \sin(\lambda_0 t)\cos(\lambda_0 t)\sin(\lambda_0 k) \\ &= \cos(\lambda_0 k)\left[\sin^2(\lambda_0 t) + \cos^2(\lambda_0 t)\right] \\ &= \cos(\lambda_0 k), \end{aligned}$$

for all $k \in [1-t, N-t]$ and $t \in [1, N]$ where N is the length of the sequence of observations.

Now, let us extend this to a d-dimensional positional encoding $\boldsymbol{e}_t$:

$$\boldsymbol{e}_t \cdot \boldsymbol{e}_{t+k} = \sum_{m=0}^{\frac{d}{2}-1} \sin(\lambda_m t)\sin(\lambda_m(t+k)) + \cos(\lambda_m t)\cos(\lambda_m(t+k)).$$

Using the same logic as in the 2-dimensional case, we find:

$$\boldsymbol{e}_t \cdot \boldsymbol{e}_{t+k} = \sum_{m=0}^{\frac{d}{2}-1} \cos(\lambda_m k),$$

for all $k \in [1-t, N-t]$ and $t \in [1, N]$ where N is the length of the sequence of observations.

Further:

$$\boldsymbol{e}_t \cdot \boldsymbol{e}_{t-k} = \sum_{m=0}^{d/2-1} \cos\left(-\lambda_m k\right).$$

Since cosine is symmetric around 0:

$$\begin{aligned}\boldsymbol{e}_t \cdot \boldsymbol{e}_{t-k} &= \sum_{m=0}^{d/2-1} \cos\left(-\lambda_m k\right) \\ &= \sum_{m=0}^{d/2-1} \cos\left(\lambda_m k\right) \\ &= \boldsymbol{e}_t \cdot \boldsymbol{e}_{t+k}.\end{aligned}$$

for all $k \in [0, N-t]$ and $t \in [1, N]$.

□

Theorem 3 (Geometric progression of the wavelengths of encoding dimensions)**.** *Consider a positional encoding vector $\boldsymbol{e}_t$ of dimension d, defined as:*

$$\boldsymbol{e}_t = \left[\sin\left(\lambda_0 t\right) \quad \cos\left(\lambda_0 t\right) \quad \sin\left(\lambda_1 t\right) \quad \cos\left(\lambda_1 t\right) \quad \cdots \quad \sin\left(\lambda_{\frac{d}{2}-1} t\right) \quad \cos\left(\lambda_{\frac{d}{2}-1} t\right)\right],$$

where $\lambda_m = 10000^{\frac{-2m}{d}}$. The wavelengths form a geometric progression from 2π to $10000 \cdot 2\pi$.

Proof. Let w_m denote the wavelength of the term $\sin\left(\lambda_m t\right)$ in the positional encoding. The wavelength is given by $w_m = \lambda_m^{-1} 2\pi$.

For $\lambda_m = 10000^{\frac{-2m}{d}}$, we have:

$$w_m = 10000^{\frac{2m}{d}} \cdot 2\pi.$$

The sequence of wavelengths $w_1, w_2, \ldots, w_{\frac{d}{2}-1}$ is:

$$2\pi, 10000^{\frac{2}{d}} \cdot 2\pi, 10000^{\frac{4}{d}} \cdot 2\pi, \ldots, 10000^{1-\frac{2}{d}} \cdot 2\pi.$$

This sequence can be rewritten as:

$$2\pi, \left(10000^{\frac{2}{d}}\right)^1 \cdot 2\pi, \left(10000^{\frac{2}{d}}\right)^2 \cdot 2\pi, \ldots, \left(10000^{\frac{2}{d}}\right)^{\frac{d}{2}-1} \cdot 2\pi,$$

which is a geometric progression with a common ratio of $10000^{\frac{2}{d}}$.

For large d, $1 - \frac{2}{d} \approx 1$, confirming that the wavelengths range form a geometric progression from 2π to $10000 \cdot 2\pi$. This holds true for both sine and cosine terms.

□

A.2 Uncertainty Estimation

Theorem 4 (Relationship between Entropy and Mutual Information)**.** *Given a random variable $\mathsf{Y} \in \{1, 2, \cdots, K\}$ representing the outcome, $\vec{\mathsf{X}}$ representing the input features, $\mathcal{D}$ a dataset with observations of $\vec{\mathsf{X}}$ and Y and $\boldsymbol{\theta}$ the parameters of a model estimating the relationship between $\vec{\mathsf{X}}$ and Y. Then we have that:*

$$\mathcal{H}\left(\mathsf{p}\left(\mathsf{Y}|\vec{\mathsf{X}}, \mathcal{D}\right)\right) - \mathcal{I}\left(\mathsf{Y}, \boldsymbol{\theta}|\vec{\mathsf{X}}, \mathcal{D}\right) = \mathbb{E}_{\mathsf{P}(\boldsymbol{\theta}|\mathcal{D})} \mathcal{H}\left(\mathsf{p}\left(\mathsf{Y}|\vec{\mathsf{X}}, \boldsymbol{\theta}\right)\right),$$

where $\mathcal{H}$ is the entropy of a distribution and $\mathcal{I}$ the mutual information between two random variables.

Proof.

$$
\begin{aligned}
&\mathcal{H}\left(\mathsf{p}\left(\mathsf{Y}|\vec{\mathsf{X}},\mathcal{D}\right)\right) - \mathcal{I}\left(\mathsf{Y},\boldsymbol{\theta}|\vec{\mathsf{X}},\mathcal{D}\right) \\
&= \mathbb{E}_{\mathsf{p}\left(\mathsf{Y}|\vec{\mathsf{X}},\mathcal{D}\right)}\left[-\log \mathsf{p}\left(\mathsf{Y}|\vec{\mathsf{X}},\mathcal{D}\right)\right] - \mathbb{E}_{\mathsf{P}(\boldsymbol{\theta}|\mathcal{D})}\mathbb{E}_{\mathsf{p}\left(\mathsf{Y}|\vec{\mathsf{X}},\boldsymbol{\theta}\right)}\left[\log\left(\frac{\mathsf{p}\left(\mathsf{Y}|\vec{\mathsf{X}},\boldsymbol{\theta}\right)}{\mathsf{p}\left(\mathsf{Y}|\vec{\mathsf{X}},\mathcal{D}\right)}\right)\right] \\
&= \mathbb{E}_{\mathsf{p}\left(\mathsf{Y}|\vec{\mathsf{X}},\mathcal{D}\right)}\left[-\log \mathsf{p}\left(\mathsf{Y}|\vec{\mathsf{X}},\mathcal{D}\right)\right] - \mathbb{E}_{\mathsf{P}(\boldsymbol{\theta}|\mathcal{D})}\mathbb{E}_{\mathsf{p}\left(\mathsf{Y}|\vec{\mathsf{X}},\boldsymbol{\theta}\right)}\left[\log \mathsf{p}\left(\mathsf{Y}|\vec{\mathsf{X}},\boldsymbol{\theta}\right)\right] \\
&\quad + \mathbb{E}_{\mathsf{P}(\boldsymbol{\theta}|\mathcal{D})}\mathbb{E}_{\mathsf{p}\left(\mathsf{Y}|\vec{\mathsf{X}},\boldsymbol{\theta}\right)}\left[\log \mathsf{p}\left(\mathsf{Y}|\vec{\mathsf{X}},\mathcal{D}\right)\right] \\
&= \mathbb{E}_{\mathsf{P}(\boldsymbol{\theta}|\mathcal{D})}\mathbb{E}_{\mathsf{p}\left(\mathsf{Y}|\vec{\mathsf{X}},\boldsymbol{\theta}\right)}\left[-\log \mathsf{p}\left(\mathsf{Y}|\vec{\mathsf{X}},\boldsymbol{\theta}\right)\right] - \mathbb{E}_{\mathsf{p}\left(\mathsf{Y}|\vec{\mathsf{X}},\mathcal{D}\right)}\left[\log \mathsf{p}\left(\mathsf{Y}|\vec{\mathsf{X}},\mathcal{D}\right)\right] \\
&\quad + \sum_{k=1}^{K} \mathbb{E}_{\mathsf{P}(\boldsymbol{\theta}|\mathcal{D})}\left[\mathsf{p}\left(\mathsf{Y}=k|\vec{\mathsf{X}},\boldsymbol{\theta}\right)\right]\log \mathsf{p}\left(\mathsf{Y}=k|\vec{\mathsf{X}},\mathcal{D}\right) \\
&= \mathbb{E}_{\mathsf{P}(\boldsymbol{\theta}|\mathcal{D})}\mathbb{E}_{\mathsf{p}\left(\mathsf{Y}|\vec{\mathsf{X}},\boldsymbol{\theta}\right)}\left[-\log \mathsf{p}\left(\mathsf{Y}|\vec{\mathsf{X}},\boldsymbol{\theta}\right)\right] - \mathbb{E}_{\mathsf{p}\left(\mathsf{Y}|\vec{\mathsf{X}},\mathcal{D}\right)}\left[\log \mathsf{p}\left(\mathsf{Y}|\vec{\mathsf{X}},\mathcal{D}\right)\right] \\
&\quad + \sum_{k=1}^{K} \mathsf{p}\left(\mathsf{Y}=k|\vec{\mathsf{X}},\mathcal{D}\right)\log \mathsf{p}\left(\mathsf{Y}=k|\vec{\mathsf{X}},\mathcal{D}\right) \\
&= \mathbb{E}_{\mathsf{P}(\boldsymbol{\theta}|\mathcal{D})}\mathbb{E}_{\mathsf{p}\left(\mathsf{Y}|\vec{\mathsf{X}},\boldsymbol{\theta}\right)}\left[-\log \mathsf{p}\left(\mathsf{Y}|\vec{\mathsf{X}},\boldsymbol{\theta}\right)\right] - \mathbb{E}_{\mathsf{p}\left(\mathsf{Y}|\vec{\mathsf{X}},\mathcal{D}\right)}\left[\log \mathsf{p}\left(\mathsf{Y}|\vec{\mathsf{X}},\mathcal{D}\right)\right] \\
&\quad + \mathbb{E}_{\mathsf{p}\left(\mathsf{Y}|\vec{\mathsf{X}},\mathcal{D}\right)}\left[\log \mathsf{p}\left(\mathsf{Y}|\vec{\mathsf{X}},\mathcal{D}\right)\right] \\
&= \mathbb{E}_{\mathsf{P}(\boldsymbol{\theta}|\mathcal{D})}\mathcal{H}\left(\mathsf{p}\left(\mathsf{Y}|\vec{\mathsf{X}},\boldsymbol{\theta}\right)\right).
\end{aligned}
$$

□

Theorem 5 (Approximate maximum likelihood estimate of the precision parameters of a Dirichlet distribution)**.** *Given a random variable $\vec{\Pi}$ which follows a Dirichlet distribution parameterised by the mean $\boldsymbol{m}$ and precision s, we can define its distribution as $\mathtt{Dir}\,(s\boldsymbol{m})$. Given the estimates for the mean $\hat{\boldsymbol{m}}$ and the observed $\boldsymbol{\pi}$, the maximum likelihood estimate of the precision can be approximated using Stirling's approximation as:*

$$\hat{s} = \frac{K-1}{2\sum_{k=1}^{K}\hat{m}_k\left(\log\hat{m}_k - \log\pi_k\right)}.$$

Proof. The log likelihood of the parameter s is given by:

$$
\begin{aligned}
\mathcal{L}(s|\boldsymbol{\pi}) &= \log \mathsf{P}\left(\vec{\Pi}=\boldsymbol{\pi}|\hat{\boldsymbol{m}},s\right) \\
&= \log\left(\frac{\Gamma(s)}{\prod_{j=1}^{K}\Gamma(sm_j)}\prod_{k=1}^{K}\pi_k^{sm_k-1}\right) \text{ (Equation 3.4)} \\
&= \log\Gamma(s) - \sum_{k=1}^{K}\log\Gamma(sm_k) + \sum_{k=1}^{K}(sm_k-1)\log\pi_k \\
&\approx \log\left(\sqrt{\frac{2\pi}{s}}\left(\frac{s}{e}\right)^{s}\right) - \sum_{k=1}^{K}\log\left(\sqrt{\frac{2\pi}{sm_k}}\left(\frac{sm_k}{e}\right)^{sm_k}\right) + \sum_{k=1}^{K}(sm_k-1)\log\pi_k \\
&\qquad\qquad \text{(Stirling's Approximation)}
\end{aligned}
$$

$$
\begin{aligned}
&= \frac{1}{2}\log(2\pi) - \frac{1}{2}\log(s) + s\log(s) - s - \frac{1}{2}\sum_{k=1}^{K}\log(2\pi) + \frac{1}{2}\sum_{k=1}^{K}\log(s) + \frac{1}{2}\sum_{k=1}^{K}\log(m_k) \\
&\quad - \sum_{k=1}^{K} sm_k \log sm_k + s\sum_{k=1}^{K} m_k + \sum_{k=1}^{K}(sm_k - 1)\log\pi_k \\
&\propto \frac{K-1}{2}\log(s) + s\log(s) - \sum_{k=1}^{K} sm_k \log sm_k + \sum_{k=1}^{K} sm_k \log\pi_k
\end{aligned}
$$

Then:

$$
\begin{aligned}
\frac{\partial\mathcal{L}(s|\boldsymbol{\pi})}{\partial s} &\approx \frac{K-1}{2s} + \frac{s}{s} + \log s - \sum_{k=1}^{K} m_k \log(s) - \sum_{k=1}^{K} m_k \log(m_k) - \sum_{k=1}^{K}\frac{sm_k}{sm_k}m_k \\
&\quad + \sum_{k=1}^{K} m_k \log\pi_k \\
&= \frac{K-1}{2s} - \sum_{k=1}^{K} m_k \log(m_k) + \sum_{k=1}^{K} m_k \log\pi_k
\end{aligned}
$$

We know that the $\hat{s}$ which maximises this likelihood will be the $\hat{s}$ such that the derivative is equal to zero. Hence:

$$
\begin{aligned}
0 &= \frac{K-1}{2\hat{s}} - \sum_{k=1}^{K} m_k \log(m_k) + \sum_{k=1}^{K} m_k \log\pi_k \\
\therefore \frac{K-1}{2\hat{s}} &= \sum_{k=1}^{K} m_k \left(\log(m_k) + \log\pi_k\right) \\
\therefore \hat{s} &= \frac{K-1}{2\sum_{k=1}^{K}\hat{m}_k\left(\log\hat{m}_k - \log\pi_k\right)}.
\end{aligned}
$$

□

Bibliography

Bahdanau, Dzmitry, Kyunghyun Cho, and Yoshua Bengio (2015). "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*. URL: http://arxiv.org/abs/1409.0473.

Bengio, Yoshua, Réjean Ducharme, Pascal Vincent, and Christian Jauvin (2003). "A Neural Probabilistic Language Model". In: *Journal of Machine Learning Research* 3.Feb, pp. 1137–1155. URL: https://dl.acm.org/doi/10.5555/944919.944966.

Bland, Amy and Alexandre Schaefer (2012). "Different Varieties of Uncertainty in Human Decision-Making". In: *Frontiers in Neuroscience* 6, p. 85. DOI: 10.3389/fnins.2012.00085.

Blundell, Charles, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra (2015). "Weight Uncertainty in Neural Networks". In: *Proceedings of the 32nd International Conference on Machine Learning (ICML) Volume 37*, pp. 1613–1622. URL: https://dl.acm.org/doi/abs/10.5555/3045118.3045290.

Bojar, Ondřej et al. (2017). "Findings of the 2017 Conference on Machine Translation (WMT17)". In: *Proceedings of the Second Conference on Machine Translation*. Association for Computational Linguistics. DOI: 10.18653/v1/W17-4717. URL: https://aclanthology.org/W17-4717.

Brown, Tom et al. (2020). "Language Models are Few-shot Learners". In: *Advances in Neural Information Processing Systems*. Vol. 33, pp. 1877–1901. URL: https://proceedings.neurips.cc/paper/2020/hash/1457c0d6bfcb4967418bfb8ac142f64a-Abstract.html.

Budzianowski, Paweł, Tsung-Hsien Wen, Bo-Hsiang Tseng, Iñigo Casanueva, Stefan Ultes, Osman Ramadan, and Milica Gašić (2018). "MultiWOZ - A Large-Scale Multi-Domain Wizard-of-Oz Dataset for Task-Oriented Dialogue Modelling". In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pp. 5016–5026. DOI: 10.18653/v1/D18-1547. URL: https://aclanthology.org/D18-1547.

Casper, Stephen et al. (2023). "Open Problems and Fundamental Limitations of Reinforcement Learning from Human Feedback". In: *arXiv preprint arXiv:2307.15217*. URL: https://arxiv.org/abs/2307.15217.

Chen, Jiuhai and Jonas Mueller (2023). "Quantifying Uncertainty in Answers from any Language Model via Intrinsic and Extrinsic Confidence Assessment". In: *arXiv preprint arXiv:2308.16175*. URL: https://arxiv.org/abs/2308.16175.

Chen, Lingjiao, Matei Zaharia, and James Zou (2023). "How is ChatGPT's Behavior Changing Over Time?" In: *arXiv preprint arXiv:2307.09009*. URL: https://arxiv.org/abs/2307.09009.

Cho, Kyunghyun, Bart van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio (2014). "On the Properties of Neural Machine Translation: Encoder–Decoder Approaches". In: *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*. Doha, Qatar: Association for Computational Linguistics, pp. 103–111. DOI: 10.3115/v1/W14-4012. URL: https://aclanthology.org/W14-4012.

Cohn, David A, Zoubin Ghahramani, and Michael I Jordan (1996). "Active Learning with Statistical Models". In: *Journal of Artificial Intelligence Research (JAIR)* 4, pp. 129–145. URL: http://mlg.eng.cam.ac.uk/pub/pdf/CohGhaJor94a.pdf.

Collins, Katherine Maeve, Matthew Barker, Mateo Espinosa Zarlenga, Naveen Raman, Umang Bhatt, Mateja Jamnik, Ilia Sucholutsky, Adrian Weller, and Krishnamurthy Dvijotham (2023). "Human Uncertainty in Concept-Based AI Systems". In: *Proceedings of the 2023 AAAI/ACM Conference on AI, Ethics, and Society*, pp. 869–889. URL: https://dl.acm.org/doi/abs/10.1145/3600211.3604692.

Desai, Shrey and Greg Durrett (2020). "Calibration of Pre-trained Transformers". In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 295–302. URL: https://aclanthology.org/2020.emnlp-main.21.

Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova (2019). "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)*. Minneapolis, Minnesota: Association for Computational Linguistics, pp. 4171–4186. DOI: 10.18653/v1/N19-1423. URL: https://aclanthology.org/N19-1423.

Eric, Mihail et al. (2020). "MultiWOZ 2.1: A Consolidated Multi-Domain Dialogue Dataset with State Corrections and State Tracking Baselines". In: *Proceedings of the 12th Language Resources and Evaluation Conference*. Marseille, France: European Language Resources Association, pp. 422–428. ISBN: 979-10-95546-34-4. URL: https://aclanthology.org/2020.lrec-1.53.

Freeman, Linton C (1965). *Elementary Applied Statistics: For Students in Behavioural Science*. Wiley.

Gal, Yarin (2016). "Uncertainty in Deep Learning". PhD thesis. University of Cambridge. URL: https://mlg.eng.cam.ac.uk/yarin/thesis/thesis.pdf.

Gal, Yarin and Zoubin Ghahramani (2016). "Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning". In: *Proceedings of the 33rd International Conference on International Conference on Machine Learning*. Vol. 3, pp. 1651–1660. URL: https://proceedings.mlr.press/v48/gal16.

Gal, Yarin, Riashat Islam, and Zoubin Ghahramani (2017). "Deep Bayesian Active Learning with Image Data". In: *International Conference on Machine Learning*. PMLR, pp. 1183–1192. URL: https://proceedings.mlr.press/v70/gal17a.

Garnelo, Marta, Dan Rosenbaum, Christopher Maddison, Tiago Ramalho, David Saxton, Murray Shanahan, Yee Whye Teh, Danilo Rezende, and S. M. Ali Eslami (2018). "Conditional Neural Processes". In: *Proceedings of the 35th International Conference on Machine Learning*. Ed. by Jennifer Dy and Andreas Krause. Vol. 80. Proceedings of Machine Learning Research. PMLR, pp. 1704–1713. URL: https://proceedings.mlr.press/v80/garnelo18a.html.

Goodfellow, Ian, Yoshua Bengio, and Aaron Courville (2016). *Deep Learning*. MIT Press. URL: http://www.deeplearningbook.org.

Grote, Thomas and Philipp Berens (2020). "On the Ethics of Algorithmic Decision-Making in Healthcare". In: *Journal of Medical Ethics* 46.3, pp. 205–211. URL: https://jme.bmj.com/content/46/3/205.abstract.

Guo, Chuan, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger (2017). "On Calibration of Modern Neural Networks". In: *Proceedings of the 34th International Conference on Machine Learning*, pp. 1321–1330. URL: https://proceedings.mlr.press/v70/guo17a.html.

Han, Ting, Ximing Liu, Ryuichi Takanabu, Yixin Lian, Chongxuan Huang, Dazhen Wan, Wei Peng, and Minlie Huang (2021). "MultiWOZ 2.3: A Multi-domain Task-Oriented Dialogue Dataset Enhanced with Annotation Corrections and Co-Reference Annotation". In: *CCF International Conference on Natural Language Processing and Chinese Computing*. Springer, pp. 206–218. URL: https://www.springerprofessional.de/en/multiwoz-2-3-a-multi-domain-task-oriented-dialogue-dataset-enhan/19743634.

Heck, Michael, Christian Geishauser, Hsien-chin Lin, Nurul Lubis, Marco Moresi, Carel van Niekerk, and Milica Gašić (Dec. 2020a). "Out-of-Task Training for Dialog State Tracking Models". In: *Proceedings of the 28th International Conference on Computational Linguistics*. Barcelona, Spain (Online): International Committee on Computational Linguistics, pp. 6767–6774. DOI: 10.18653/v1/2020.coling-main.596. URL: https://aclanthology.org/2020.coling-main.596.

Heck, Michael, Nurul Lubis, Carel van Niekerk, Shutong Feng, Christian Geishauser, Hsien-Chin Lin, and Milica Gašić (2022). "Robust Dialogue State Tracking with Weak Supervision and Sparse Data". In: *Transactions of the Association for Computational Linguistics* 10, pp. 1175–1192. DOI: 10.1162/tacl_a_00513. URL: https://aclanthology.org/2022.tacl-1.68.

Heck, Michael, Carel van Niekerk, Nurul Lubis, Christian Geishauser, Hsien-Chin Lin, Marco Moresi, and Milica Gašić (2020b). "TripPy: A Triple Copy Strategy for Value Independent Neural Dialog State Tracking". In: *Proceedings of the 21th Annual Meeting of the Special Interest Group on Discourse and Dialogue*. Association for Computational Linguistics, pp. 35–44. URL: https://www.aclweb.org/anthology/2020.sigdial-1.4.

Henderson, Matthew, Blaise Thomson, and Jason D. Williams (2014a). "The Second Dialog State Tracking Challenge". In: *Proceedings of the 15th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*. Philadelphia, PA, U.S.A.: Association for Computational Linguistics, pp. 263–272. DOI: 10.3115/v1/W14-4337. URL: https://aclanthology.org/W14-4337.

Henderson, Matthew, Blaise Thomson, and Steve Young (2013). "Deep Neural Network Approach for the Dialog State Tracking Challenge". In: *Proceedings of the SIGDIAL 2013 Conference*. Metz, France: Association for Computational Linguistics, pp. 467–471. URL: https://www.aclweb.org/anthology/W13-4073.

Henderson, Matthew, Blaise Thomson, and Steve Young (2014b). "Robust Dialog State Tracking using Delexicalised Recurrent Neural Networks and Unsupervised Adaptation". In: *Proceedings of the 2014 IEEE Spoken Language Technology Workshop (SLT)*. IEEE, pp. 360–365. DOI: 10.1109/SLT.2014.7078601.

Hinton, Geoffrey, Nitish Srivastava, and Kevin Swersky (2012). *Neural Networks for Machine Learning. Lecture 6A: Overview of Mini-Batch Gradient Descent*. URL: http://www.cs.toronto.edu/~hinton/coursera/lecture6/lec6.pdf.

Hinton, Geoffrey, Oriol Vinyals, and Jeffrey Dean (2015). "Distilling the Knowledge in a Neural Network". In: *Conference on Neural Information Processing Systems (NIPS) - Deep Learning and Representation Learning Workshop*. URL: http://arxiv.org/abs/1503.02531.

Hirsh, Jacob B, Raymond A Mar, and Jordan B Peterson (2012). "Psychological Entropy: A Framework for Understanding Uncertainty-Related Anxiety." In: *Psychological Review* 119.2, p. 304. URL: https://psycnet.apa.org/record/2012-00550-001.

Houlsby, Neil, Ferenc Huszar, Zoubin Ghahramani, and Máté Lengyel (2011). "Bayesian Active Learning for Classification and Preference Learning". In: *arXiv preprint arXiv:1112.5745 Version 1*. URL: http://arxiv.org/abs/1112.5745v1.

Iovine, Andrea, Pasquale Lops, Fedelucio Narducci, Marco de Gemmis, and Giovanni Semeraro (2022). "An Empirical Evaluation of Active Learning Strategies for Profile Elicitation in a Conversational Recommender System". In: *Journal of Intelligent Information Systems* 58.2, pp. 337–362. ISSN: 1573-7675. DOI: 10.1007/s10844-021-00683-4.

Johansson, Ulf, Tuve Lofstrom, and Lars Niklasson (2007). "The Importance of Diversity in Neural Network Ensembles - An Empirical Investigation". In: *2007 International Joint Conference on Neural Networks*, pp. 661–666. DOI: 10.1109/IJCNN.2007.4371035.

Kim, Sungdong, Sohee Yang, Gyuwan Kim, and Sang-Woo Lee (2020). "Efficient Dialogue State Tracking by Selectively Overwriting Memory". In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 567–582.

Kingma, Diederik P and Jimmy Ba (2017). "Adam: A Method for Stochastic Optimization". In: *arXiv preprint arXiv:1412.6980 Version 9*. URL: https://arxiv.org/abs/1412.6980v9.

Krishnan, Ranganath, Mahesh Subedar, and Omesh Tickoo (2020). "Specifying Weight Priors in Bayesian Deep Neural Networks with Empirical Bayes". In: *Proceedings of the AAAI Conference on Artificial Intelligence* 34.04, pp. 4477–4484. DOI: 10.1609/aaai.v34i04.5875. URL: https://ojs.aaai.org/index.php/AAAI/article/view/5875.

Lakshminarayanan, Balaji, Alexander Pritzel, and Charles Blundell (2017). "Simple and Scalable Predictive Uncertainty Estimation using Deep Ensembles". In: *Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS 2017)*. Long Beach, California, USA.

Lecun, Y., L. Bottou, Y. Bengio, and P. Haffner (1998). "Gradient-Based Learning Applied to Document Recognition". In: *Proceedings of the IEEE* 86.11, pp. 2278–2324. DOI: 10.1109/5.726791.

Lee, Harrison, Samrat Phatale, Hassan Mansoor, Kellie Lu, Thomas Mesnard, Colton Bishop, Victor Carbune, and Abhinav Rastogi (2023). "RLAIF: Scaling Reinforcement Learning from Human Deedback with AI Feedback". In: *arXiv preprint arXiv:2309.00267*. URL: https://arxiv.org/pdf/2309.00267.pdf.

Lee, Hwaran, Jinsik Lee, and Tae-Yoon Kim (2019). "SUMBT: Slot-Utterance Matching for Universal and Scalable Belief Tracking". In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Florence, Italy: Association for Computational Linguistics, pp. 5478–5483. DOI: 10.18653/v1/P19-1546. URL: https://aclanthology.org/P19-1546.

Levin, Esther, Roberto Pieraccini, and Wieland Eckert (1998). "Using Markov Decision Process for Learning Dialogue Strategies". In: *Proceedings of the 1998 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP'98 (Cat. No. 98CH36181)*. Vol. 1. IEEE, pp. 201–204. URL: https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=674402.

Li, Shiyang, Semih Yavuz, Kazuma Hashimoto, Jia Li, Tong Niu, Nazneen Rajani, Xifeng Yan, Yingbo Zhou, and Caiming Xiong (2020). "CoCo: Controllable Counterfactuals for Evaluating Dialogue State Trackers". In: *International Conference on Learning Representations (ICLR)*. URL: https://arxiv.org/abs/2010.12850.

Lin, Weizhe, Bo-Hsiang Tseng, and Bill Byrne (2021a). "Knowledge-Aware Graph-Enhanced GPT-2 for Dialogue State Tracking". In: *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. Online and Punta Cana, Dominican Republic: Association for Computational Linguistics. DOI: 10.18653/v1/2021.emnlp-main.620. URL: https://aclanthology.org/2021.emnlp-main.620.

Lin, Zhaojiang et al. (2021b). "Leveraging Slot Descriptions for Zero-Shot Cross-Domain Dialogue State Tracking". In: *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics. DOI: 10.18653/v1/2021.naacl-main.448. URL: https://aclanthology.org/2021.naacl-main.448.

Liu, Yinhan et al. (2019). "RoBERTa: A Robustly Optimized BERT Pretraining Approach". In: *arXiv preprint arXiv:1907.11692 Version 1*. URL: https://arxiv.org/abs/1907.11692v1.

Loshchilov, Ilya and Frank Hutter (2019). "Decoupled Weight Decay Regularization". In: *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. URL: https://openreview.net/forum?id=Bkg6RiCqY7.

Luhn, H. P. (1957). "A Statistical Approach to Mechanized Encoding and Searching of Literary Information". In: *IBM Journal of Research and Development* 1.4, pp. 309–317. DOI: 10.1147/rd.14.0309. URL: https://ieeexplore.ieee.org/abstract/document/5392697.

Luong, Thang, Hieu Pham, and Christopher D. Manning (Sept. 2015). "Effective Approaches to Attention-based Neural Machine Translation". In: *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Lisbon, Portugal: Association for Computational Linguistics, pp. 1412–1421. DOI: 10.18653/v1/D15-1166. URL: https://aclanthology.org/D15-1166.

Malinin, Andrey (2019). "Uncertainty Estimation in Deep Learning with application to Spoken Language Assessment". PhD thesis. University of Cambridge. URL: https://www.repository.cam.ac.uk/handle/1810/298857.

Metallinou, Angeliki, Dan Bohus, and Jason D. Williams (2013). "Discriminative state tracking for spoken dialog systems". In: *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics. URL: http://research.microsoft.com/apps/pubs/default.aspx?id=192335.

Mikolov, Tomas, Kai Chen, Greg Corrado, and Jeffrey Dean (2013). "Efficient estimation of word representations in vector space". In: *arXiv preprint arXiv:1301.3781*. URL: https://arxiv.org/abs/1301.3781.

Mrkšić, Nikola, Diarmuid Ó Séaghdha, Tsung-Hsien Wen, Blaise Thomson, and Steve Young (2017). "Neural Belief Tracker: Data-Driven Dialogue State Tracking". In: *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Vancouver, Canada:

Association for Computational Linguistics, pp. 1777–1788. DOI: 10.18653/v1/P17-1163. URL: https://www.aclweb.org/anthology/P17-1163.

Nair, Vinod and Geoffrey E Hinton (2010). "Rectified Linear Units Improve Restricted Boltzmann Machines". In: *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pp. 807–814. DOI: 10.5555/3104322.3104425.

Narayan, Sridhar (1997). "The Generalized Sigmoid Activation Function: Competitive Supervised Learning". In: *Information Sciences* 99.1, pp. 69–82. ISSN: 0020-0255. DOI: 10.1016/S0020-0255(96)00200-9. URL: https://doi.org/10.1016/S0020-0255(96)00200-9.

Neal, Radford M (2012). *Bayesian Learning for Neural Networks*. Vol. 118. Springer Science & Business Media. URL: https://link.springer.com/book/10.1007/978-1-4612-0745-0.

OpenAI (2023). *GPT-4 Technical Report*. Tech. rep. URL: https://arxiv.org/abs/2303.08774.

Osband, Ian, Zheng Wen, Seyed Mohammad Asghari, Vikranth Dwaracherla, Morteza Ibrahimi, Xiuyuan Lu, and Benjamin Van Roy (2023). "Epistemic Neural Networks". In: *arXiv preprint arXiv:2107.08924*. URL: https://arxiv.org/abs/2107.08924.

Ouyang, Long et al. (2022). "Training Language Models to Follow Instructions with Human Feedback". In: vol. 35, pp. 27730–27744. URL: https://proceedings.neurips.cc/paper_files/paper/2022/hash/b1efde53be364a73914f58805a001731-Abstract-Conference.html.

Pennington, Jeffrey, Richard Socher, and Christopher Manning (2014). "GloVe: Global Vectors for Word Representation". In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, pp. 1532–1543. DOI: 10.3115/v1/D14-1162. URL: https://aclanthology.org/D14-1162.

Peters, Matthew E., Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer (2018). "Deep Contextualized Word Representations". In: *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. Association for Computational Linguistics, pp. 2227–2237. DOI: 10.18653/v1/N18-1202. URL: https://aclanthology.org/N18-1202.

Qian, Ning (1999). "On the Momentum Term in Gradient Descent Learning Algorithms". In: *Neural Networks* 12.1, pp. 145–151. ISSN: 0893-6080. DOI: https://doi.org/10.1016/S0893-6080(98)00116-6. URL: https://www.sciencedirect.com/science/article/pii/S0893608098001166.

Radford, Alec, Jong Wook Kim, Tao Xu, Greg Brockman, Christine Mcleavey, and Ilya Sutskever (2023). "Robust Speech Recognition via Large-Scale Weak Supervision". In: *Proceedings of the 40th International Conference on Machine Learning*. Ed. by Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett. Vol. 202. Proceedings of Machine Learning Research. PMLR, pp. 28492–28518. URL: https://proceedings.mlr.press/v202/radford23a.html.

Radford, Alec, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever (2018). "Improving Language Understanding by Generative Pre-Training". In: URL: http://openai-assets.s3.amazonaws.com/research-covers/language-unsupervised/language_understanding_paper.pdf.

Radford, Alec, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever (2019). "Language Models are Unsupervised Multitask Learners". In: URL: https://cdn.openai.com/better-language-models/language_models_are_unsupervised_multitask_learners.pdf.

Raffel, Colin, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu (2020). "Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer". In: *The Journal of Machine Learning Research* 21.1. ISSN: 1532-4435. URL: https://dl.acm.org/doi/abs/10.5555/3455716.3455856.

Rastogi, Abhinav, Xiaoxue Zang, Srinivas Sunkara, Raghav Gupta, and Pranav Khaitan (2020). "Towards Scalable Multi-Domain Conversational Agents: The Schema-Guided Dialogue Dataset". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 34. 05, pp. 8689–8696. URL: https://ojs.aaai.org/index.php/AAAI/article/view/6394.

Ren, Liliang, Kaige Xie, Lu Chen, and Kai Yu (2018). "Towards Universal Dialogue State Tracking". In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Brussels,

Belgium: Association for Computational Linguistics, pp. 2780–2786. DOI: 10.18653/v1/D18-1299. URL: https://aclanthology.org/D18-1299.

Robert, Christian P and George Casella (1999). *Monte Carlo Statistical Methods*. Vol. 2. Springer. URL: https://link.springer.com/book/10.1007/978-1-4757-4145-2.

Rumelhart, David E. and James L. McClelland (1987). "Learning Internal Representations by Error Propagation". In: *Parallel Distributed Processing: Explorations in the Microstructure of Cognition: Foundations*, pp. 318–362. URL: https://ieeexplore.ieee.org/document/6302929.

Ryabinin, Max, Andrey Malinin, and Mark Gales (2021). "Scaling Ensemble Distribution Distillation to Many Classes with Proxy Targets". In: *Advances in Neural Information Processing Systems* 34, pp. 6023–6035. URL: https://proceedings.neurips.cc/paper/2021/file/2f4ccb0f7a84f335affb418aee08a6df-Paper.pdf.

Schütze, Hinrich (1998). "Automatic Word Sense Discrimination". In: *Computational Linguistics* 24.1, pp. 97–123. URL: https://www.aclweb.org/anthology/J98-1004.

Sener, Ozan and Silvio Savarese (2018). "Active Learning for Convolutional Neural Networks: A Core-Set Approach". In: *International Conference on Learning Representations*. URL: https://arxiv.org/abs/1708.00489.

Shan, Yong, Zekang Li, Jinchao Zhang, Fandong Meng, Yang Feng, Cheng Niu, and Jie Zhou (2020). "A Contextual Hierarchical Attention Network with Adaptive Objective for Dialogue State Tracking". In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Online: Association for Computational Linguistics, pp. 6322–6333. DOI: 10.18653/v1/2020.acl-main.563. URL: https://www.aclweb.org/anthology/2020.acl-main.563.

Shannon, Claude E (1948). "A Mathematical Theory of Communication". In: *The Bell System Technical Journal* 27.3, pp. 379–423.

Shelmanov, Artem et al. (Apr. 2021). "Active Learning for Sequence Tagging with Deep Pre-trained Models and Bayesian Uncertainty Estimates". In: *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*. Online: Association for Computational Linguistics, pp. 1698–1712. DOI: 10.18653/v1/2021.eacl-main.145. URL: https://aclanthology.org/2021.eacl-main.145.

Shumailov, Ilia, Zakhar Shumaylov, Yiren Zhao, Yarin Gal, Nicolas Papernot, and Ross Anderson (2023). "The Curse of Recursion: Training on Generated Data Makes Models Forget". In: *arXiv preprint arxiv:2305.17493*. URL: https://arxiv.org/abs/2305.17493.

Sparck Jones, Karen (1972). "A Statistical Interpretation of Term Specificity and its application in Retrieval". In: *Journal of Documentation* 28.1, pp. 11–21. ISSN: 0022-0418. DOI: 10.1108/eb026526. URL: https://doi.org/10.1108/eb026526.

Stanovich, Keith E (2009). *What Intelligence Tests Miss: The Psychology of Rational Thought*. Yale University Press. URL: https://psycnet.apa.org/record/2010-12180-012.

Szandała, Tomasz (2021). "Review and Comparison of Commonly Used Activation Functions for Deep Neural Networks". In: *Bio-inspired Neurocomputing*, p. 203. URL: https://link.springer.com/content/pdf/10.1007/978-981-15-5495-7.pdf#page=208.

Szegedy, Christian, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna (2016). "Rethinking the Inception Architecture for Computer Vision". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2818–2826. DOI: 10.1109/CVPR.2016.308.

Touvron, Hugo et al. (2023). "LLaMA: Open and Efficient Foundation Language Models". In: *arXiv preprint arXiv:2302.13971*. URL: https://arxiv.org/abs/2302.13971.

van Niekerk, Carel, Christian Geishauser, Michael Heck, Shutong Feng, Hsien-chin Lin, Nurul Lubis, Benjamin Ruppik, Renato Vukovic, and Milica Gašić (2023). "CAMELL: Confidence-based Acquisition Model for Efficient Self-supervised Active Learning with Label Validation". In: *arXiv preprint arXiv:2310.08944 Version 1*. URL: https://arxiv.org/abs/2310.08944.

van Niekerk, Carel, Michael Heck, Christian Geishauser, Hsien-chin Lin, Nurul Lubis, Marco Moresi, and Milica Gašić (2020). "Knowing What You Know: Calibrating Dialogue Belief State Distributions via Ensembles". In: *Findings of the Association for Computational Linguistics: EMNLP 2020*. Online: Association for Computational Linguistics, pp. 3096–3102. DOI: 10.18653/v1/2020.findings-emnlp.277. URL: https://www.aclweb.org/anthology/2020.findings-emnlp.277.

van Niekerk, Carel, Andrey Malinin, Christian Geishauser, Michael Heck, Hsien-chin Lin, Nurul Lubis, Shutong Feng, and Milica Gašić (2021). "Uncertainty Measures in Neural Belief Tracking and the Effects on Dialogue Policy Performance". In: *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. Online and Punta Cana, Dominican Republic: Association for Computational Linguistics. URL: https://aclanthology.org/2021.emnlp-main.623.

Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin (2017). "Attention Is All You Need". In: *Advances in neural information processing systems*, pp. 5998–6008. URL: https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

Vinyals, Oriol, Charles Blundell, Timothy Lillicrap, Daan Wierstra, et al. (2016). "Matching Networks For One Shot Learning". In: *Advances in neural information processing systems* 29, pp. 3630–3638. URL: https://proceedings.neurips.cc/paper_files/paper/2016/file/90e1357833654983612fb05e3ec9148c-Paper.pdf.

Williams, Jason D. (June 2014). "Web-style Ranking and SLU Combination for Dialog State Tracking". In: *Proceedings of the 15th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*. Philadelphia, PA, U.S.A.: Association for Computational Linguistics, pp. 282–291. DOI: 10.3115/v1/W14-4339. URL: https://aclanthology.org/W14-4339.

Williams, Jason D. and Steve Young (2007). "Partially observable Markov decision processes for spoken dialog systems". In: *Computer Speech & Language* 21.2, pp. 393–422. ISSN: 0885-2308. DOI: https://doi.org/10.1016/j.csl.2006.06.008. URL: https://www.sciencedirect.com/science/article/pii/S0885230806000283.

Xiong, Miao, Zhiyuan Hu, Xinyang Lu, Yifei Li, Jie Fu, Junxian He, and Bryan Hooi (2023). "Can LLMs Express Their Uncertainty? An Empirical Evaluation of Confidence Elicitation in LLMs". In: *arXiv preprint arXiv:2306.13063*. URL: https://arxiv.org/abs/2306.13063.

Xiong, Wayne, Jasha Droppo, Xuedong Huang, Frank Seide, Michael L. Seltzer, Andreas Stolcke, Dong Yu, and Geoffrey Zweig (2017). "Toward Human Parity in Conversational Speech Recognition". In: *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 25.12, pp. 2410–2423. DOI: 10.1109/TASLP.2017.2756440.

Ye, Fanghua, Jarana Manotumruksa, and Emine Yilmaz (Sept. 2022). "MultiWOZ 2.4: A Multi-Domain Task-Oriented Dialogue Dataset with Essential Annotation Corrections to Improve State Tracking Evaluation". In: *Proceedings of the 23rd Annual Meeting of the Special Interest Group on Discourse and Dialogue*. Edinburgh, UK: Association for Computational Linguistics, pp. 351–360. URL: https://aclanthology.org/2022.sigdial-1.34.

Young, Steve, Milica Gai, Simon Keizer, François Mairesse, Jost Schatzmann, Blaise Thomson, and Kai Yu (2010). "The Hidden Information State model: A practical framework for POMDP-based spoken dialogue management". In: *Computer Speech & Language* 24.2, pp. 150–174. ISSN: 0885-2308. DOI: https://doi.org/10.1016/j.csl.2009.04.001. URL: https://www.sciencedirect.com/science/article/pii/S0885230809000230.

Young, Steve, Jost Schatzmann, Karl Weilhammer, and Hui Ye (2007). "The Hidden Information State Approach to Dialog Management". In: *2007 IEEE International Conference on Acoustics, Speech and Signal Processing - ICASSP '07*. Vol. 4, pp. IV–149–IV–152. DOI: 10.1109/ICASSP.2007.367185.

Zang, Xiaoxue, Abhinav Rastogi, Srinivas Sunkara, Raghav Gupta, Jianguo Zhang, and Jindong Chen (2020). "MultiWOZ 2.2 : A Dialogue Dataset with Additional Annotation Corrections and State Tracking Baselines". In: *Proceedings of the 2nd Workshop on Natural Language Processing for Conversational AI*. Online: Association for Computational Linguistics, pp. 109–117. DOI: 10.18653/v1/2020.nlp4convai-1.13. URL: https://aclanthology.org/2020.nlp4convai-1.13.

Zhang, Jianguo, Kazuma Hashimoto, Chien-Sheng Wu, Yao Wang, Philip Yu, Richard Socher, and Caiming Xiong (Dec. 2020). "Find or Classify? Dual Strategy for Slot-Value Predictions on Multi-Domain Dialog State Tracking". In: *Proceedings of the Ninth Joint Conference on Lexical and Computational Semantics*. Barcelona, Spain (Online): Association for Computational Linguistics, pp. 154–167. URL: https://aclanthology.org/2020.starsem-1.17.