Design and Development of Spoken Dialogue System in Indic Languages

A Thesis Submitted

In Partial Fulfilment of the Requirements for the Degree

of

DOCTOR OF PHILOSOPHY

Submitted by

Shrikant Malviya

Under the Supervision of

Prof. U. S. Tiwary

to the

DEPARTMENT OF INFORMATION TECHNOLOGY

INDIAN INSTITUTE OF INFORMATION TECHNOLOGY ALLAHABAD

PRAYAGRAJ-211015 (U.P.)-INDIA

March 2022

INDIAN INSTITUTE OF INFORMATION TECHNOLOGY

ALLAHABAD

(A Centre of Excellence in Information Technology Established by Govt. of India)

CANDIDATE DECLARATION

I, **Shrikant Malviya**, Enrolment No. **RS162** certify that this thesis work entitled **"Design and Development of Spoken Dialogue System in Indic Languages"** is submitted by me in partial fulfilment of the requirement of the Degree of **Ph.D.** in Department of **Information Technology, Indian Institute Of Information Technology, Allahabad**.

I understand that plagiarism includes:

1. Reproducing someone else's work (fully or partially) or ideas and claiming it as one's own.
2. Reproducing someone else's work (Verbatim copying or paraphrasing) without crediting.
3. Committing literary theft (copying some unique literary construct).

I have given due credit to the original authors/sources through proper citation for all the words, ideas, diagrams, graphics, computer programs, experiments, results, websites that are not my original contribution. I have used quotation marks to identify verbatim sentences and given credit to the original authors/sources.

I affirm that no portion of my work is plagiarised. In the event of a complaint of plagiarism, I shall be fully responsible. I understand that my Supervisor may not be in a position to verify that this work is not plagiarised.

Name: Shrikant Malviya **Date:** March 25, 2022

Enrolment No.: RS162

Department: Information Technology

Place: IIITA, Prayagraj

Abstract

Language is a system of words used as symbols to convey ideas, while dialogue is a natural and effective method for users to interact with and access information from other humans or machines. In recent years, substantial improvements in speech recognition performance have enticed the research community to build natural conversational interfaces in the form of a Spoken Dialogue System (SDS). This work is concerned broadly with designing a complete spoken dialogue system in an *Indic language* scenario, i.e. Hindi. No significant work has been done earlier to promote the research and development of a *Hindi spoken dialogue system*. Hence, it becomes critical for the thesis to address the issues and challenges unveiled for the *Hindi* language through introducing new datasets, methods and measures to build and evaluate all the integral modules of the Hindi SDS.

A typical SDS structure is based upon a modular pipeline design connecting five principal components in a specific order: Automatic Speech Recognition (ASR), Spoken Language Understanding (SLU) and/or Dialogue State Tracking (DST), Dialogue Management (DM), Natural Language Dialogue Generation (NLDG) and Text-To-Speech (TTS) synthesiser. The work presented in this thesis demonstrates how these components are developed individually and integrated to develop a real-world spoken dialogue system in Hindi.

As the Hindi text contains lots of lexical/morphological ambiguities, therefore, it becomes a key challenge for SLU/DST and NLDG models to appropriately detect the Dialogue-Act (DA), understand the utterances and generate natural responses. Hindi is very rich in inflectional morphology. There is usually a limit of 8-9 inflected word forms of nouns in English, but in Hindi, it is more than 40. The way a language is spoken and written gets changed from place to place. It leads to the introduction of variations where the meaning of a sentence is the same, but the way to express it gets changed.

Other language-related challenges that a Hindi SDS have to deal with are code-mixing, hidden information, echo-words, etc. Code-mixing is the mixing of more languages in the conversation. There are many cases in the corpus where the user had expressed some words from English during the conversation. (Example: "मुझे कम रेंज वाले रेस्तरां की तलाश है।" (I am looking for low (cost) range restaurants.)). Here the word "रेंज" (range) is an English word that gives an indication of the cost. Therefore, in the belief state tracking, the word "कम" (less) needs to be associated with costing after the resolution of the codemix. Hidden-Information is prevalent in conversation

in that the people do not convey each and everything they need; instead, they give an indication that makes it more interesting.

The thesis starts with a discussion on *language understanding* and its challenges concerning the Hindi language. This module aims to translate user input into an accurate representation of the *user goal* in the form of DAs, which helps in keeping *track* of the dialogue state. A *dialogue state* is a Markovian representation that denotes a full representation of the information received from the user at a point in time. This module has to deal with certain challenges, i.e. modelling of linguistic variation, speech recognition errors and the influence of dialogue contexts. In order to understand the research questions that underlie the SLU and DST module in the perspective of Indic languages (Hindi), we collect a dialogue corpus: Hindi Dialogue Restaurant Search (HDRS) corpus and compare various state-of-the-art SLU and DST models on it. Conceptualisation of SLU and DST as a single module, the DST models based on traditional embeddings, i.e. Word2Vec, GloVe and FastText based as well as recent BERT-based embeddings are explored to show how they are able to deal with challenges of Hindi language, i.e. morphological/lexical ambiguities, code-mix words, echo words and hidden information which exist in the utterances.

The *Dialogue Management* (DM) infer the current dialogue state to take the appropriate action. The dialogue manager is often modelled as a *Reinforcement Learning* (RL) task, enabling the system to learn to act optimally by maximising a reward function for goal-oriented applications. Describing the limitations of traditional methods of policy learning, i.e. value-based and policy-based methods such as low sample-efficiency, high variance and often converge to local optima, the thesis investigates sample-efficient deep-learning RL methods, which resolve the issues by applying actor-critic algorithms with *experience replay* mechanism.

The *Natural Language Dialogue Generation* (NLDG) is a critical component as it significantly impacts the usability and perceived quality of a spoken dialogue system. Rule-based (or template-based) NLDG systems are widely utilised due to their simplicity, robustness and high accuracy in limited domains. However, the repetition of identical responses makes the dialogue tedious and bored for most real-world users. Moreover, such systems also suffer from scalability issues to large domains. To investigate data-driven methods of Hindi NLDG, the thesis presents a natively collected dataset of *unstructured input-output pair* of dialogue-act (system's) and corresponding natural response. Later, the Recurrent Neural Network Language Generation (RNNLG) framework based models, along with their analysis of how they extract intended meaning in terms of *content planning* (modelling semantic input) and *surface realisation* (final sentence generation) are experimented on the proposed unaligned Hindi dataset.

For *speech synthesisers* as a last component in the dialogue pipeline, we not only train several TTS systems but also propose a *quality assessment framework* to evaluate them. The TTS models, i.e. Unit selection speech synthesis (USS), Hidden Markov Model speech synthesis (HMM), Clustergen speech synthesis (CLU)

and Deep Neural Network-based speech synthesis (DNN), are considered to obtain the synthesised speech on two publicly available Hindi speech datasets, i.e. CMU-Indic Database, IITM Indic-TTS Database. Describing the limitations of conventional subjective and objective evaluation measures, a novel method of quality assessment, a *Learning-Based Objective Evaluation* (LBOE), is proposed, which utilises a set of selected low-level-descriptors (LLD) based features to analyse the speech-quality of TTS models.

Overall, the work presented in this thesis, in the form of presented corpora and proposed methods, makes steps towards building more flexible real-world spoken dialogue systems in Indic languages.

Keywords: Spoken dialogue systems, dialogue corpus in Indic languages, dialogue management, dialogue state tracking, natural language dialogue generation, text-to-speech, data-driven models.

Acknowledgements

Gratitude is first due to all my teachers who have guided me throughout my life, for without them, it would not have been possible to reach this milestone in my life. Specifically, I would like to thank my supervisor, Prof. U. S. Tiwary, for his devoted supervision throughout my PhD at the Indian Institute of Information Technology Allahabad. His guidance has been invaluable for my research. He has always found time to meet me and respond to my questions, and for that, I am very grateful to him.

The SILP Lab research group has provided a highly motivating and pleasant environment to carry out research. For that and all the support they have given me, I must thank the current and previous group members: Gyanendra K Verma, Santosh Kumar Barnwal, Sudhakar Mishra, Rohit Mishra, Punit Singh, Varsha Singh, Mohd. Asif and Sumit Singh. I am grateful to Rohit Mishra for his excellent support in building the Indic language Dialogue corpora. It has been an incredible journey.

I would like to express my special thanks to 'Ministry of Education' for scholarship. I would also like to thank the Cambridge Dialogue Systems Group researchers for making open-source a plethora of their work that inspired and motivated me to carry out dialogue research on the Indic language.

Lastly, I would like to offer a sincere thank to my parents, R. P. Malviya and Shobha Malviya, for their unfailing support and encouragement. I would also like to thank my cousin brother Vaibhav Malviya, my family and my wife; I could not have finished this thesis without their support.

At last, I want to present my special thank to almighty God. My singular thank to all the people who always motivated and supported directly or indirectly during my PhD work.

Shrikant Malviya

March 25, 2022

IIITA, Prayagraj

Dedicated

to

my teachers and family

Contents

List of Figures

List of Tables

List of Symbols

Language Understanding & State Tracking

$\mathbf{b}_t$	Output of belief-updation layer at turn t
$\mathbf{c}_t$	Encoded candidate slot-value pair at turn t
$\mathbf{f}_t^{v,cnn}$	CNN feature vector at turn t
$\mathbf{s}_t$	Encoded system response at turn t
$\mathbf{u}_t$	Encoded user utterance at turn t
$\mathbf{y}_t$	Output of decision-making layer at turn t
$\mathcal{L}$	Estimated loss
$\oplus$	Concatenation operator
$\overleftarrow{h}$	Hidden state of B-LSTM
$\overrightarrow{h}$	Hidden state of F-LSTM
$\psi(\cdot)$	Representing a modelling function
$\sigma(\cdot)$	Sigmoid activation function
$\text{CNN}_{s,v}^{(\cdot)}(\cdot)$	slot-value specific delexicalised CNN feature extractor
$\text{encode}(\cdot)$	An encoder function, i.e. GLAD, GCE, GSAT
$\log(\cdot)$	Log function
C_*	Self-attention context of an input obtained from the encoder
$exp(\cdot)$	Exponential function $e^{(\cdot)}$
g_t^v	Pre-softmax activation for value v of slot s at turn t
H_*	Hidden encoding of an input obtained from the encoder
m_t	System utterance at turn t
p_t^ϕ	Probability that the slot s is not mentioned by the user upto turn t.
p_t^v	Probability of value v expressed for slot s at turn t
u_t	User utterance at turn t
lr	Learning rate

Modelling Dialogue Management

α	Learning rate

γ	Geometric discount constant
$\mathbb{1}(\mathcal{D})$	Indicator function to denote the success of a dialogue
$\mathcal{GP}(m(\cdot,\cdot), k(\cdot,\cdot))$	Gaussian Process (GP) with $m(\cdot,\cdot)$ and $k(\cdot,\cdot)$ as mean and kernel functions
$\mathcal{L}(\cdot)$	Cross-entropy based loss function
∇_*	Gradient estimated for a policy with parameter '*'
$\nabla_\theta J(\cdot)$	Estimated gradient of expected reward
Ω	Set of observations
$\pi(\cdot)$	Policy function
ρ	Important Sampling (IS) ratio
$\tau \sim \pi$	A dialogue trajectory τ generated through policy π
τ	Trajectory of an entire dialogue
$\mathbf{b}(s)$	Belief state distribution
θ	Parameter of policy-based RL method
$\widehat{\nabla_\theta J(\theta)}$	Total expected reward
A	Set of actions
$A_*(\cdot,\cdot)$	Advantage function
c	A hyperparameter denoting upper bound for IS weight
$E_\pi(\cdot)$	Expectation function for policy π
$J(\cdot)$	Expected reward on a model, e.g. policy (π), parameters (θ)
n	A hyperparameter denoting number of training steps per episode
O	Observation probability function
$Q^*(\cdot,\cdot)$	Q function
R	Reward function
S	Set of states
T	Markovian state-transition probability function
$V^*(\cdot)$	Value function

Natural Language Generation

$\beta_{*,*}$	Estimated attention-weight
η	Regularisation constant set to 10^{-4}
ξ	Regularisation constant set to 100
$\mathbf{d}_*$	Encoded dialogue act
$\mathbf{h}_*$	Hidden state of the model
$\mathbf{i}_*, \mathbf{f}_*, \mathbf{o}_*$	input, forget and output gates of LSTM-cell

$\mathbf{w}_*$	Embedding of a token w_t
$\mathbf{W}_{*,*}$	Model parameters
$\mathcal{L}(\cdot)$	Cross-entropy loss
$\odot$	Element-wise multiplication operator
$\omega_{*,*}$	0,1 normalised attention-weight
$\oplus$	Concatenation operator
$\sigma(\cdot)$	Sigmoid activation function
$\tanh(\cdot)$	Tangent Hyperbolic activation function
$\mathrm{C_m}$	Verb post-modifier in a sentence
$\mathrm{O_m}$	Object modifier in a sentence
$\mathrm{S_m}$	Subject modifier in a sentence
$\mathrm{V_m}$	Verb modifier in a sentence
V_s	A set of slot-specific values
$\mathrm{S_{inf}}$	List of informable slots
$\mathrm{S_{req}}$	List of requestable slots

Quality Assessment of Synthesised Speech

$\boldsymbol{\eta}$	Features for Cross-Validation
$\hat{\boldsymbol{\beta}}$	Parameter vector of assessment model
$\hat{\mathcal{Y}}$	Estimated quality of speech
$\mathcal{P}_{syn}$	Physical property attributes of synthesised speech
$\mathcal{S}_{syn}$	Synthesised speech signal
$\mathcal{Y}$	Actual subjective quality of speech
ν	A SVM parameter which specifies number of support-vectors
$f_{\hat{\boldsymbol{\beta}}}(\cdot)$	Approximation function for quality assessment
$R(\cdot,\cdot)$	RMSE function

Chapter 1

Introduction

Since the computing systems came into existence, the idea of *talking to machines* has enticed the entire world and inspired, keep motivated generations of researchers. According to the Oxford dictionary, a *dialogue* is "a conversation between two or more people". The same dictionary defines a *computer* as "an electronic device which is capable of receiving information (data) in a particular form" and able "to produce a result in the form of information or signals". The question arises, is dialogue only restricted to humans, as mentioned in the Oxford dictionary, or can it be considered as an exchange of information as human-like communication, for example, via speech where computers can also take part one day? This thesis focusses on developing techniques in order to build such computer systems in Indic language that can hold a conversation using natural language just like humans do among themselves.

A *Spoken Dialogue System* (SDS) is a computer system that can maintain an intelligent conversation with a human via speech in real-time [2]. Developing such systems has been a longstanding goal of artificial intelligence (AI). In this thesis, the emphasis is placed on developing a real-world spoken dialogue system in an Indic language, i.e. Hindi, which can establish a conversational with some practical use on a large enough domain. Furthermore, the thesis not only addresses the issues of scaling these systems to large domains but also resolves language-related challenges as this is the first attempt to build a "Hindi spoken dialogue system".

An autonomous system capable of holding a conversation with a human has numerous applications and implications. They vary to a vast range from deep philosophical aspects such as exploring the nature of consciousness and self-awareness to more objective aspects such as large-scale automation of menial labour in call centres, technical support and many other industries. For many human-computer interactions, communication through natural language presents an ideal interface as it is fully expressive and requires no user training.

The Turing Test (1950) is established as one of the most influential criteria for *machine intelligence* that measures the autonomous dialogue agent's ability to masquerade as a human being [3]. In 1966, Joseph Weizenbaum designed ELIZA as an attempt to pass the Turing test [4]. It was a template-based dialogue agent designed to

talk to users about their personal problems. Based on *person-centred* psychotherapy, ELIZA aimed to provide an empathic environment for the patient to explore the answers to their problems through introspection rather than by offering expert advice:

ELIZA : आप कैसे हैं? मुझे अपनी समस्या बताएं!

USER : मैं फिलहाल थोड़ा परेशान हूँ।

ELIZA : क्या आप अक्सर ऐसे परेशान रहते हैं?

USER : कभी–कभी

ELIZA : और बताइये।

USER : मुझे अपने दोस्तों से परेशानी है।

ELIZA : क्या आप अच्छे दोस्त हैं?

This example is taken from our Hindi implementation[1] of ELIZA.

By designing ELIZA, Weizenbaum tried to show how superficial human-machine communication would be - it is made up of less than 200 lines of code. But he was astonished by the number of persons who convinced and felt human-like emotions in talking with ELIZA; even his assistant spent hours talking to the agent, revealing her most intimate problems. In fact, they believed that there was a human psychotherapist on the other side of the interface, despite knowing that the ELIZA is nothing but a sequence of meticulously written template rules that made it a good impulsive listener.

With the onset of the Fourth Industrial Revolution[2] and adoption of new technologies such as smartphones, homes, and others, conversational agents, e.g., Apple's Siri, Microsoft's Cortana and Amazon's Alexa, are permeating into every aspect of human life and allow users to achieve a plethora of tasks using their voice, e.g. playing music or movies, scheduling meetings, switching on/off room's light, and many others. However, most of these agents only focus on textual (or voice) modality, performing simple tasks and answering factual questions [5].

Research [6] has shown that in many rural areas of developing countries, i.e. India, more people regularly use mobile phones than can read or write. Availability of automated systems provides the facility to access useful information such as weather and agriculture reports. For some time, people were significantly interested in building an open-domain dialogue system that can handle arbitrary conversations. Ideally, such a system would understand and respond in the same way as a human might do but has encyclopaedic knowledge. However, building such a system has proved challenging [7], and the current state-of-the-art is still very far away.

[1] Hindi-ELIZA: https://github.com/skmalviya/ELIZA-AndroidApp

[2] According to Klaus Schwab of the World Economic Forum: "*This Fourth Industrial Revolution is, however, fundamentally different. It is characterised by a range of new technologies that are fusing the physical, digital and biological worlds, impacting all disciplines, economies and industries, and even challenging ideas about what it means to be human.*"

As it became clear that building open-domain dialogue agents would be no easy feat, the focus of research shifted to a bottom-up paradigm. Instead of trying to mimic humans in general conversation, task-oriented dialogue systems could help users accomplish specific, well-defined tasks, such as flight reservation system [8], tourist guidance [9], movie searches [10], troubleshooting domain [11], and information retrieval [12]. In terms of usability and reliability, such systems have to be resilient and be able to deal with different types of users while remaining relatively easy to develop and extend [13]. With this idea, the goal of the thesis is to design and develop a task-oriented spoken dialogue system in an Indic language such as Hindi.

The remainder of the chapter discusses the general perspective of the thesis in constructing a native SDS. Section 1.1 introduces the structure of an SDS and its basic components. Further, Section 1.2 emphasises the motivation behind work done in the thesis. The aims and objectives of the work are pointed out in Section 1.3. Finally, the contributions and the thesis outline is presented in Section 1.4, including the publications.

1.1 Spoken Dialogue System Structure

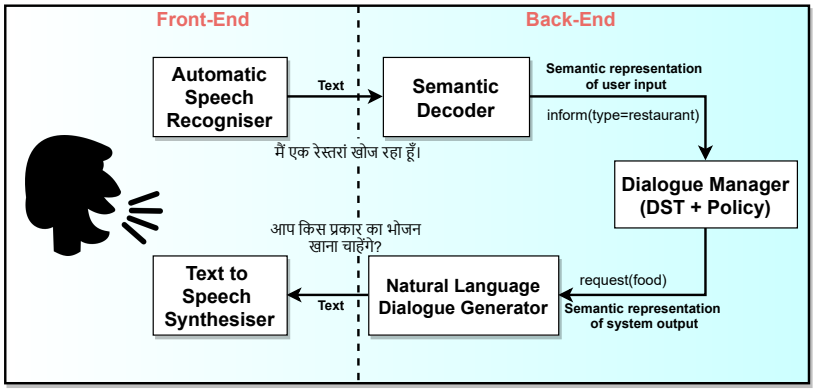


Figure 1.1 Spoken dialogue system structure.

Spoken dialogue is a conversational exchange between two or more people where the primary medium is speech. To simplify the task for a dialogue system, two assumptions are made [14]. First, the conversation will take place only between two participants in a dialogue. Second, the dialogue will be considered as a sequence of *turns*, where each turn consists of two utterances: one by the user, the other by the dialogue system, in a fixed order.

There is no consensus in the literature on the architecture of a spoken dialogue system. However, a statistical SDS can be thought of a pipeline design [15] connecting five principal components in a specific order, as shown in Figure 1.1. One cycle through the pipeline completes one *dialogue turn* such that one utterance from each participant: the user and the dialogue system.

The first component in the SDS pipeline is the *Automatic Speech Recognition* (ASR), which transcribes the user's speech into text. For the last two decades, speech recognition has been dominated by traditional statisti-

cal approaches such as Hidden Markov Model (HMM) combined with feedforward Artificial Neural Network (ANN) [16]. Currently, a deep learning method, i.e. Long Short-Term Memory (LSTM) [17], has taken over many aspects of speech recognition. In 2015, Google's speech recognition achieved a remarkable performance jump of 49% through Connectionist Temporal Classification (CTC) [18] trained LSTM, which is now available through Google Voice Search on mobile as well as computer.

The text obtained from the ASR module is then forwarded to the next module a *semantic decoder*, which performs the task of *Spoken Language Understanding* (SLU). This module's goal is to determine the underlying semantics of the given utterance. It broadly covers the research related to domain detection, intent determination [19, 20], and slot-filling [21, 22]. In dialogue research, the semantics are typically represented in a standard form: *Dialogue-Act*[3] (DA). For example, the utterance:

'मैं एक रेस्तरां खोज रहा हूँ।'

can be formed as:

`inform(type=restaurant)`

Once the user's intent has been determined in the form of dialogue-act, the *Dialogue Management's* (DM) role comes into play. This module has two jobs to accomplish. First is the job of *Dialogue State Tracking* (DST), which accumulates or updates any previous information conveyed during the conversation in a *dialogue-state*[4] [24]. Second is the job of *dialogue policy* to choose an appropriate reply to the user based on the updated dialogue state in the current turn. The most notable approaches for learning a dialogue policy are based on Reinforcement Learning (RL) [2], which view the dialogue interaction as a long-term planning task and optimise its action selection policy to achieve a higher success rate. The system reply is again in the form of a dialogue act (semantic units), e.g. `request(food)`.

The dialogue manager's output is then passed to the next component in the pipeline: the *Natural Language Dialogue Generation* (NLDG). It transforms the abstract semantics notation of the system-act back into a text representation. For example, the dialogue act:

`request(food)`

can be transformed to:

'आप किस प्रकार का भोजन खाना चाहेंगे? '.

Initially, rule (template)-based approaches [25, 26] or a hybrid of handcrafted and statistical methods [27–29] were used to build NLDG systems in an SDS. However, they are restricted to semantically-aligned corpora, which are tedious and expensive to build. On achieving success on machine-translation and language-modelling [30, 31], the neural-network-based models are successfully applied in generating natural system utterances in a spoken dialogue [12, 32, 33].

[3] DA is derived from the concept of the speech act [23], representing the meaning of an utterance.

[4] The Markovian representation, which summarises the current state of the dialogue, is called the *dialogue state*.

At last, the Text-To-Speech (TTS) component converts the NLDG component's output into its audio representation. Early TTS systems were based on the concatenative approach, i.e. *unit selection*, concatenating pre-recorded speech fragments (from a single speaker) into audio output representing the given word sequence [34]. The unit selection approach has since been superseded by the *parametric* approach [35], where the model is mainly parametrised by a Hidden Markov Model (HMM). Further, gaining success of *neural network models*, the deep and recurrent neural networks have also been successfully applied to parametric speech synthesis [36–38].

Following the modular architecture to build an SDS in Indic language, we plan to work on developing each component separately and resolve the challenges in combining them in a single SDS pipeline design later. The overall research is focussed on constructing all the system components as statistical models with parameters learned directly from the data by resolving various language-specific and language-independent challenges. Hence, the research moves sequentially with the investigation and implementation of various SDS components, i.e. ASR, SLU, DST, DM, NLDG and TTS. The research also incorporates the phases of data collection and building new models as required during the development.

1.2 Motivation

As discussed earlier, an SDS is typically build upon various components: a speech recogniser, a semantic decoding module for spoken language understanding, a dialogue manager for dialogue state tracking and policy learning, a language generation module, and a speech synthesiser. This thesis is concerned broadly with designing a complete spoken dialogue system in an *Indic language* scenario, i.e. Hindi. No significant work is done earlier to promote the research and development of a *Hindi spoken dialogue system*. Hence, it becomes critical for the thesis to address the issues and challenges unveiled for the *Hindi* language through introducing new datasets, methods and measures to build and evaluate all the integral modules of the Hindi SDS.

In a statistical spoken dialogue system, the aim is to replace each of the aforementioned components with a statistical model with parameters estimated from data [39, 40]. The overall goal is to build a data-driven dialogue system with the ability to be get improved over time and be perceived as behaving human-like by the users. The components of such systems are based on statistical methods, i.e. probabilistic distribution, neural network models, which allow them to handle uncertainty in both their input and their output [2, 41].

As the Hindi text contains lots of lexical/morphological ambiguities, therefore, it becomes a key challenge for DST and NLDG models to appropriately detect the DAs, understand the utterances and generate natural responses. Hindi is very rich in inflectional morphology. There is usually a limit of 8-9 inflected word forms of nouns in English [42], but in Hindi, it is more than 40 [43, 44]. The way a language is spoken and written

changes from place to place. It leads to the introduction of variations where the meaning of a sentence is the same, but the way to express it gets changed [45].

Other language-related challenges that a Hindi SDS have to deal with are code-mixing [46], hidden information [47], echo-words [48], etc. *Code-mixing* is the mixing of more languages in the conversation. There are many cases in the corpus where the user had expressed some words from English during the conversation. (Example: "मुझे कम रेंज वाले रेस्तरां की तलाश है।" ("I am looking for low range restaurants.")). Here the word "रेंज" (range) is an English word that gives an indication of the cost. Therefore, in the belief state tracking, the word "कम" (less) need to be associated with costing after the resolution of the codemix [46, 49]. *Hidden-Information* is prevalent in conversation that the people do not convey each and everything they need; instead, they give an indication, which makes it more interesting [47].

1.3 Aims and Objectives

In this work, We aim to design and develop a virtual assistant able to process both spoken as well as written utterances and provide information related to an application domain and able to conversate with the user in Indic language, i.e. **Hindi**. To this end, we divide the work to be accomplished into several phases:

The following are the aims and objectives of the work:

- To collect and release a Hindi dialogue corpus containing a large number of labelled dialogues for the experiments.
- To provide the details of features, collection process and statistical analysis of the proposed corpus.
- To show the performance of the state-of-the-art models for SLU, DST, DM and NLDG tasks.
- To incorporate the Advantage Actor-Critic with Experience Replay (A2CER) algorithm for dialogue policy learning which has recently been shown to be performing well on simple gaming environments.
- To investigate and propose a new RNNLG-based (Recurrent Neural Network Language Generation) model on a natively developed corpus.
- To train and build various TTS systems on publicly available datasets from scratch.
- To propose a novel framework that explores the generalisation capabilities of low-level descriptor-based perceptual features and investigates to what extent they can be used to measure the synthetic speech quality at all without subjective testing.
- To develop an integrated web-based interface to a dialogue agent named "SILPAssistant" (SILPA[5]).

[5]SILPA (SILPAssistant) acquired its name on the acronym of our Lab's name SILP (Speech, Image & Langauge Processing) Lab

1.4 Contributions & Thesis Outline

The dissertation is organised as follows:

- **Chapter 1 - Introduction**

  Current Chapter 1 presents the introduction to the work explored in the thesis. After describing the usability and effectiveness of the dialogue systems in our day-to-day life, the chapter depicts the basic pipelined structure of a statistical SDS. Then, the underlying motivation and challenges behind developing a real-world SDS in Indic language Hindi are briefed. At last, the chapter presents contributions and an outline of the thesis.

- **Chapter 2 - Challenges & Research Gap in Hindi SDS**

  The general concepts and architecture of an SDS are described in this chapter, with detailed descriptions of the challenges and research gap observed in its development, which lays the foundations of statistical spoken dialogue systems and sets the context of the work.

- **Chapter 3 - HDRS: Language Understanding & State Tracking**

  This chapter raises the key research questions that underlie the SLU and DST module in building a Hindi dialogue system for the restaurant domain. To conduct the research, an indigenously developed corpus, Hindi Dialogue Restaurant Search (HDRS), is proposed and compared various state-of-the-art SLU and DST models. The chapter also signifies the conceptualisation of SLU and DST as a single module. The DST models based on traditional embedding, i.e. Word2Vec, GloVe and FastText based as well as recent BERT[6]-based embeddings are explored to show how they are able to deal with the Hindi language challenges of morphological/lexical ambiguities, code-mix words, echo words and hidden information exist in the utterances. Part of the research work has been published in the following publications [1, 50]:

  S. Malviya, R. Mishra, S. K. Barnwal, and U. S. Tiwary, "HDRS: Hindi dialogue restaurant search corpus for dialogue state tracking in task-oriented environment", In IEEE/ACM Transactions on Audio, Speech, and Language Processing 2021.

  D. Goswami, S. Malviya, R. Mishra, U.S. Tiwary, "Analysis of Word-level Embeddings for Indic Languages on AI4Bharat-IndicNLP Corpora", In IEEE 8th Uttar Pradesh Section International Conference on Electrical, Electronics and Computer Engineering (UPCON) IEEE, 2021.

- **Chapter 4 - Modelling Dialogue Management through Reinforcement Learning**

  This chapter is concerned with learning a dialogue policy that determines which action to take given the current state of the dialogue. The chapter first reviews the background knowledge and summarises related

[6]BERT: Bidirectional Encoder Representations from Transformers

works on dialogue policy optimisation and reward estimation, with a specific focus on RL approaches. Describing the limitations of traditional methods of policy learning, i.e. *value-based* and *policy-based* methods such as low sample-efficiency, high variance and often converge to local optima, the chapter investigates sample-efficient deep-learning methods, which resolve the issues by applying *actor-critic* methods with *experience replay*. This research work has been published in the following paper [51]:

Shrikant Malviya, Piyush Kumar, Suyel Namasudra, and Uma Shankar Tiwary. "Experience replay based deep reinforcement learning for dialogue management optimisation", ACM transactions of low-resource language information processing, 2022.

- **Chapter 5 - Hindi Dialogue Generation**

  This chapter investigates the Natural Language Dialogue Generation (NLDG) module by exploring the data-driven methods that generate system responses with indented attributes like fluency, variation, readability, scalability, and adequacy in the Hindi language. The first obstacle in building a data-driven model is tackled by collecting and releasing a corpus of unstructured input-output pair of dialogue-act (system's) and corresponding natural response. The chapter then presents some Recurrent Neural Network Language Generation (RNNLG) framework based models along with their analysis of how they extract intended meaning in terms of *content planning* (modelling semantic input) and *surface realisation* (final sentence generation) on the proposed unaligned Hindi dataset. This research work has been presented in the publication [52]:

  S. Singh, S. Malviya, R. Mishra, S. K. Barnwal, U. S. and Tiwary, "RNN based language generation models for a Hindi dialogue system" In International Conference on Intelligent Human-Computer Interaction, pages 124–137. Springer, 2019.

- **Chapter 6 - Quality Assessment of Synthesised Speech**

  This chapter discusses not only the different types of speech synthesisers but also compare them on various methods of evaluating TTS systems with the proposed evaluation framework: *LBOE (Learning-Based Objective Evaluation)*. The chapter first presents the working of Unit selection speech synthesis (USS), Hidden Markov Model speech synthesis (HMM), Clustergen speech synthesis (CLU) and Deep Neural Network-based speech synthesis (DNN) methods to construct speech synthesis models on two Hindi speech datasets. Traditional evaluation methods such as subjective and objective evaluation methods. Some of the contributions has been accepted for publication as [53]:

  S. Malviya, R. Mishra, S. K. Barnwal and U. S. Tiwary, "A framework for quality assessment of synthesised speech using learning-based objective evaluation" International Journal of Speech Technology.

In summary, the thesis contributions are divided into chapters corresponding to each component of the SDS. Chapter 2 describes the modular architecture of the SDS with the details of recent advancements in the development of each module with specified challenges and research gaps. Chapter 3 demonstrates our contribution of HDRS corpus and a comparison of several DST state-of-the-art models on it. Chapter 4 discusses the use of reinforcement learning in building dialogue policy modules with the help of synthetic data generated by a user-simulator. The experiments with RNN-based NLDG modules on indigenously collected Hindi corpus are depicted in Chapter 5. Chapter 6 builds several TTS systems in Hindi and also proposes a framework for quality assessment of the synthesised speech. Finally, the conclusions, limitations and future directions of the work presented in this thesis are discussed in Chapter 7.

Chapter 2

Challenges & Research Gap in Hindi SDS

This chapter provides an overview of Spoken Dialogue System (SDS) and their core components, mentioning recent advancements in their development for highlighting the challenges and research gap in a Hindi spoken dialogue system.

2.1 Modular Spoken Dialogue Systems

We design our Hindi SDS by dividing it into five modules in a pipeline architecture [15] and connecting them in a specific order, as shown in Figure 2.1. One cycle through the pipeline completes one *dialogue turn* such that one utterance from each participant: the user and the dialogue system. Hence, the overall research in the end-to-end *statistical dialogue system* focusses on constructing all the system components as statistical models with parameters learned directly from the data [40]. The remainder of the section demonstrates the characteristics of task-oriented dialogue systems, the domain ontologies and the taxonomy of the dialogue acts incorporated in designing them.

2.1.1 Task-Oriented Dialogue Systems

This thesis focusses on exploring and implementing a task-oriented dialogue system[1] in a native Indic language. The objective of such dialogue systems is to interact with the user and provide information related to a certain *application-domain* based on a large database. Some of them include flight booking systems [54], restaurant-finding system [55] or tourist information system [56].[2]

The slots are another important constituent of *slot-filling-based* dialogue systems, which not only define the dialogue domain but also specify all the tasks the system can help the users with. In more detail, the slots decide

[1] This term is used interchangeably with *goal-oriented* dialogue systems.

[2] Alternative chat-bot style systems do not make use of task ontologies or the pipeline model. Instead, these models learn to generate/choose system responses based on previous dialogue turns [57].

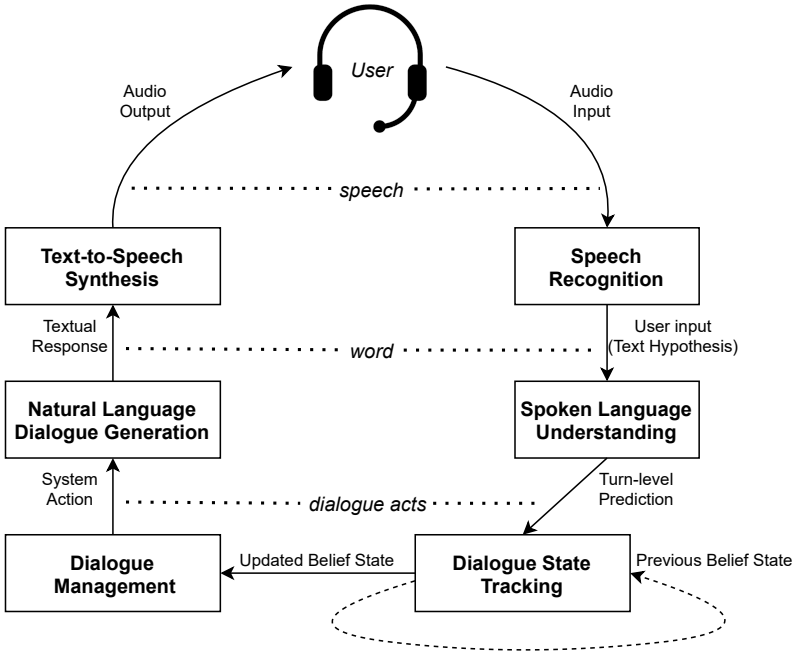


Figure 2.1 The pipeline of the core components in statistical spoken dialogue systems.

all possible actions the system can take, the possible semantics from the user's utterances and achievable states by the system [58].

2.1.2 Domain Ontologies

The *domain ontology* is made up of domain-specific slots. It has sufficient information to model the user's goal up to a given point during the conversation, referred to as the *dialogue state*. In general, a domain specific-ontology comprises informable slots S_{inf} and requestable slots S_{req}. For the domain used in this thesis, $S_{inf} \subseteq S_{req}$, though both sets can be disjoint as well. Figure 2.2 shows a subset of the ontology of the restaurant domain used by all the components in the development of our SILPA Dialogue system.

The database of the restaurant domain maintains a set of attributes necessary to keep all the relevant information of a restaurant. The informable-slots represent all the attributes required by the user to constrain his search during the conversation. On the other hand, requestable-slots constitute those attributes that the users can ask about but not necessarily use as search constraints. For instance, in the restaurant search scenario, a user might be able to search for restaurants by asking about a specific *food* type but may ask about the phone number or address only when the dialogue agent comes up with a restaurant suggestion.

Hence, the ontology consists of two sets: the set of requestable slots S_{req} and the set of informable slots S_{inf}, and for each $s \in S_{inf}$ a set of slot-specific values V_s. Given the ontology, the language understanding component detects the occurrence of such slots and their values in the user utterance. Further, the dialogue manager tracks these slot-value attributes and decides the following system action. Using these slot attributes in association with

database information, the NLDG component constructs the response or query in natural language, i.e. Hindi. The terminology and description of the domain studied in this thesis are explained in Appendix A.

INFORMABLE SLOTS: {
  FOOD: [
    अफ्रीकन, साउथ इंडियन, पंजाबी, कोरियन, सिंधी, इंडोनेशियन, बंगाली,
    ऑस्ट्रेलियन, मलेशियन, यूरोपियन, ब्राजीलियन, शाकाहारी, जैपनीज,
    चाइनीस, स्पेनिश, अफगान, फ्रेंच, इंटरनेशनल, मराठी, इटालियन ...
  ],
  NAME: [
    कैलाश पर्बत, कान्हा श्याम, रॉयल स्पाइस, दस्तरख्वान, जायसवाल डोसा कॉर्नर,
    द टेस्टी पॉइंट, सिल्क रेस्तरां, महाराजा रेस्ट्रो, डोमिनोस पिज़्ज़ा, द सेकंड वाइफ,
    करी किंग, जेड स्ट्रीट, पेंगुइन इंडियन एंड मुगल फूड्स, रिज़र्व सीट, फ़ूड किंग,
    मक्खनंस, हिगिंस, हांडी, सतकार रेस्तरां, फ़ूड स्क्वायर, लज़ीज़ रेस्तरां, ...
  ],
  PRICE RANGE: [
    सस्ता,
    मध्यम,
    महंगा
  ],
  AREA: [
    केंद्र,
    दक्षिण,
    पश्चिम,
    पूर्व,
    उत्तर
  ]
}
REQUESTABLE SLOTS: [
  address,
  area,
  food,
  name,
  phone,
  price range,
  postcode.
]

Figure 2.2 A part of the restaurant domain ontology for the SILPA dialogue system. The complete ontology consists of 34 values for food-slot, 118 name values, 3 values for price-range slot, 5 area values and 7 requestable slot values.

2.1.3 Dialogue Act Formalism

Dialogue acts are the semantic representation of both user utterances and system prompts. It is an internal representation of the intention conveyed in the utterance expressed by the user or system. The interaction takes place at the dialogue act level under the dialogue state tracking and dialogue management part, as shown in Figure 2.1. We utilise a semantic representation that is compact enough to keep their number relatively low and carry sufficient information to sustain the dialogue flow. The dialogue act theory is formalised in [59] and is

considered extendible to other task-oriented dialogue domains. In slot-based systems, the dialogue acts consist of a sequence of semantic information: first a *dialogue act type* and a set of *slot-value pairs* which combinely represent the *dialogue act arguments* [58]. A complete list of dialogue acts with their descriptions is given in Appendix B, used in the current work.

The dialogue act type represents the general action of the utterance. The systems investigated in this thesis consider three basic types of actions such as informing constraints, i.e. `inform(food=बंगाली)`, requesting information, i.e. `request(phone number)` and performing acknowledgement, i.e. *hi*, *thanking you* and *bye*. At any point in the conversation, the dialogue state consists of dialogue acts from both user and system utterance with the corresponding semantic information as a sequence of slot-value pairs. As an example, Figure 2.3 depicts the dialogue act information extracted from an utterance in terms of *turn-goals*, *turn-requests* and *system-actions* with updated dialogue state after each turn.

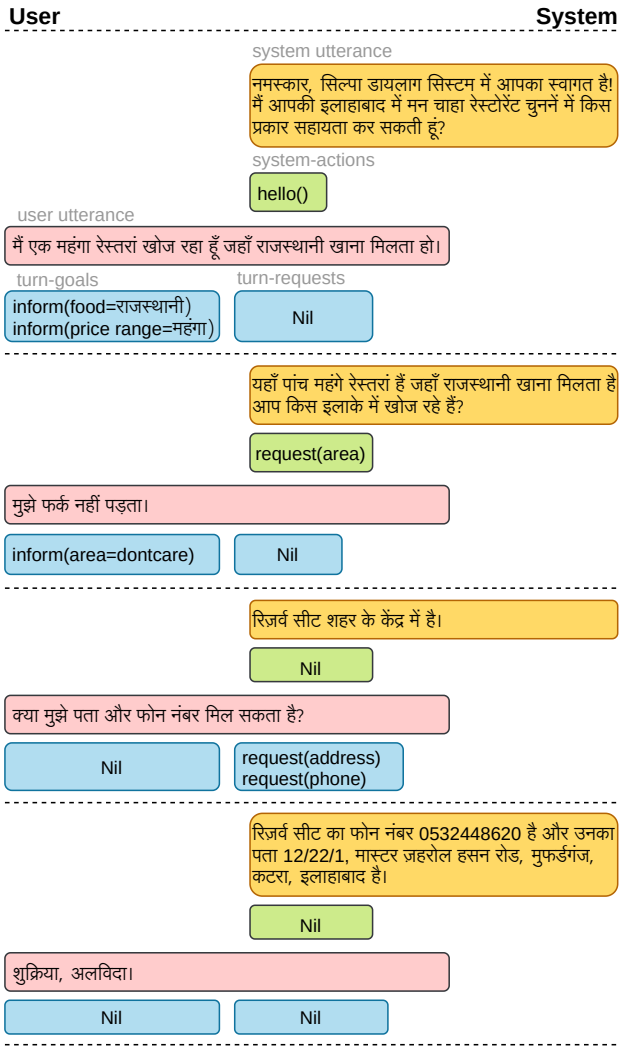


Figure 2.3 A dialogue from the HDRS corpus collected on restaurant domain. Each turn, separated by the dashed lines, contains a *system utterance* (yellow) followed by corresponding *system-actions* (green) as well as *user utterance* (red) comes with the specified *turn-goals* and *turn-requests* (blue). Box with 'Nil' entry depicts the unexpressed entity. Appendix C presents the translation of utterances expressed in the conversation.

2.1.4 Noise (uncertainty) in Dialogue

The pipeline design of the SDS (see Figure 2.1) can be considered analogous to computer networks scenario in certain aspects which facilitate noisy communication between two end-nodes [60]. Under this analogy, the two end-nodes are the user and the dialogue agent in our case. Similar to the layers in computer networks, our dialogue system has three communication layers, as shown in Figure 2.1. Each layer corresponds to a different level of abstraction:

1. **Speech**: This abstraction layer belongs to the speech interface in SDS design. During the communication, the physical sound waveform is taken as input by the ASR module and generated speech as output by the TTS module.

2. **Word**: The second layer of abstraction deals with the textual representation of user queries obtained from the ASR and passed to the language understanding module, which applies SLU and DST. Similarly, the NLDG module also produces the output in the form of natural text based on the system acts suggested by the Dialogue Management (DM) as input.

3. **Dialogue Acts**: It is a domain-specific semantic representation that provides a formal language for expressing user goals as well as the system response. On one side, this formalism help in providing an interface between the user's intents and goals (user constraints). On the other side, it also draws the required information from the external database.

Compared with computer networking, it introduces additional uncertainty into the representation of user goals as move to higher levels of abstraction in the SDS pipeline. This uncertainty is caused due to the introduction of noises at various levels of abstraction. The two primary sources of the uncertainty are: **1)** The noise introduced either by imperfect speech recognition or noisy environments, which gets progressively worse with moving ahead in the abstraction layers. **2)** the uncertainty introduced by the language understating modules due to difficulty handling lexical/morphological variations and ambiguities or understanding contextual feedback.

Overall, developing a full-fledged spoken dialogue system in Indic languages is the main focus of this thesis, which follows a modular architecture on a task-oriented scenario. We next discuss each component of the SDS design pipeline in the following sections, mentioning the challenges and research gap concerning the Hindi language.

2.2 Automatic Speech Recognition

Automatic Speech Recognition (ASR) is the first module in our pipelined SDS architecture. Its job is to transcribe the speech waveform into written form output. A typical output of an ASR module is represented as an

N-best list of the most probable hypotheses with corresponding probabilities, i.e. as a *word lattice* or *confusion network*, formed in a directed graph where each edge denotes a word and acoustic weight to estimate the path probability [61].

Most ASR systems carry an acoustic model, a language model and a set of lexicons. A typical ASR system takes raw speech waveform or a pre-processed feature vector, i.e. Mel-Frequency Cepstral Coefficients (MFCC). Based on the processed feature vectors, the acoustic model learns the most probable sequence of basic speech units, i.e. phoneme, which best represent the input vectors. The sequence of speech units generated from the previous is then mapped to texts using a lexicon. The traditional ASR systems use graphical generative models, i.e. Hidden Markov Model (HMM), to model acoustic characteristics of the speech signal, where the state probabilities are estimated through a Gaussian Mixture Model (GMM). A comprehensive review of such ASR systems is presented in Gales et al. [62]. In recent years, the research is now concentrated on using discriminative neural networks to estimate the GMM state probabilities and has became the state-of-the-art of today's ASR systems [63–65]. On the other hand, the language models help in capturing word-transition, which provides linguistic constraints to select the proper word in a sequence. Traditionally, such models are implemented using n-gram models [66, 67]. Recently, neural networks have also become state-of-the-art for learning the language models [68, 69].

We have incorporated and adapted the recent version of Google ASR in our system. To make it easily deployable, the current speech recognition research is shifted towards building *End-to-end* ASR with the aim to combine acoustic model and language model using RNNs [70–72, 69]. Currently, a deep learning method, i.e. Long Short-Term Memory (LSTM) [17], has taken over many aspects of speech recognition. In 2015, Google's speech recognition achieved a remarkable performance jump of 49% through Connectionist Temporal Classification (CTC) [18] trained LSTM, which is now available through Google Voice Search on mobile as well as computer.

2.3 Challenges in SLU & DST

2.3.1 Spoken Language Understanding (SLU)

Spoken Language Understanding (SLU), the next component in the SDS pipeline after ASR, identifies and extracts the underlying semantics of an utterance expressed by the user. Although, there are many semantic representations available to be used, most existing spoken dialogue systems use a shallow level of semantic representation called *Dialogue-Act* (DA) [73], akin to the concept of *speech act* [23]. We design the dialogue act taxonomies that capture just enough meaning in an utterance to facilitate rational dialogue behaviour within

the application domain [74]. It defines the amount of semantics it can model and hence decides the scalability and learnability of the system.

An SLU component (or a *semantic decoder*) takes a sentence as input and maps it to an output dialogue act representing underlying semantics. Overall, SLU covers the research area of domain detection (in a multi-domain scenario), intent determination [19, 20], and slot filling [21, 22]. For example, the utterance:

'मैं एक महंगा रेस्तरां खोज रहा हूँ जहाँ राजस्थानी खाना मिलता हो।'

(I am looking for an expensive restaurant where Rajsthani food is served.)

can be represented as:

`inform(type=restaurant,price range=महंगा,food=राजस्थानी).`

The obtained semantics capture not only the intent of the utterance as `action=inform`, (which means that the user is providing certain information) but also a set of actual slot-value pairs represented as the arguments of the act: `price range=महंगा`, `food=राजस्थानी`, and the application domain: `type=restaurant`. This type of semantics is generally used to represent the logic form in computational linguistic and general artificial intelligence. There are many other ways of semantic representation depending on the design of the ontology and application, such as relation-based semantic representation [75], which has richer structured concepts. In our work, only of limited domain are undertaken; hence the flat semantic representations (dialogue acts) are used. A similar form of semantics is also used as the conceptual representation of the output from the dialogue management module in Section 2.4.

Initially, many spoken dialogue systems used simple and straightforward methods such as *semantic template grammar*, e.g. Phoenix parser [76], to extract semantic details and discern the dialogue act [77]. To model more complex sentences with richer linguistic variations and ambiguities, more sophisticated and advanced grammar formalism, such as Combinatory Categorial Grammars (CCG) [78, 79], Context-Free Grammars (CFG) [80], inductive logic programming based methods [81] or parsing based methods for long-range dependencies [82, 14] have been investigated. Such domain-specific rule-based systems are hard to design and often need multiple iterations of user-testing before achieving satisfactory real-time performance and coverage [39]. Besides, these rules must be expanded or re-designed from scratch when domain converge is updated; thus they are not scalable.

For statistical SLU, it requires substantial training data with their corresponding labels [83]. Two major labelling mechanisms are used in the data-driven SLU; methods that perform the sequential labelling require *word-level* (aligned) labelling, while methods that label the entire sentence need sentence level (unaligned) labelling. The Air Travel Information System (ATIS) is a commonly used dataset where both aligned and unaligned examples are present [54]. The SLU approaches can further be divided into two categories: *generative* approaches such as Dynamic Bayesian Network (DBN) [84] and *discriminative* models such as Support Vector

Machine (SVM) [85, 86], Conditional Random Field (CRF) [87, 88]. More recently, neural network based SLU models have also been investigated [89, 90, 22, 91].

To better understand the underlying challenges in Hindi, we designed a dialogue corpus in the restaurant domain consisting of the labelled utterances in the Hindi language and incorporated a range of language understanding models on it (briefly discussed in Chapter 3).

2.3.2 Dialogue State Tracking (DST)

The term *dialogue state* roughly denotes the full-representation of the user goals at any point during the conversation. In a single-turn dialogue scenario, such as when questions in each turn are independent, the SLU output provides enough information to encode the user's request fully. However, in multi-turn dialogues, the *dialogue state* must be tracked over turns to accumulate crucial information required by the system to make decisions. For example, in a slot-filling dialogue system, the dialogue state comprises a list of constraints (goals) given by the user so far in the conversation.

Hence, we include a Dialogue State Tracking (DST) module in our system capable of accumulating the evidences appropriately in the multi-turn dialogue scenario and predicting the dialogue state effectively based on the observation and context. Probabilistic models often maintain it as the distribution over all dialogue states referred to as the *belief state* in the literature [92].

In recent years, the DST has been intensively investigated by several research groups. The completion of five successful Dialogue State Tracking Challenges (DSTC), e.g. DSTC-1 [93], DSTC-2 [94], DSTC-3 [56], DSTC-4 [95] and DSTC-5 [96], has not only increased the interest in DST but also spurred research in many dimension of the dialogue like multi-domain, multi-model scenario. These initial DSTC challenges were dedicated to cover a wide range of DST tasks, such as single and multi-domain interactions, goal-changing scenarios and human-human conversations.

Spoken Language Understanding (SLU) and Dialogue State Tracking (DST) were separate in the traditional dialogue system pipeline. The former utilises hand-crafted rules such as *basic* or *focus* trackers [58] to update the dialogue state given the output from the SLU component, and the latter learns the tracking task from the given data. These SLU-detached trackers are prone to accumulating errors received from the SLU module. Subsequently, research on belief trackers gets focussed on conceptualising SLU and DST as a single module [97–99]. To achieve better generalisation, these trackers rely on hand-crafted *semantic-dictionaries* and *delexicalisation*. In research works [100, 24], Convolutional Neural Network (CNN) based representation learning was applied to learn relevant features from semantically-induced word embeddings to predict each state stochastically without relying on hand-crafted features. Dealing with multi-domain scenario (MultiWOZ dataset [101]), many neural

Table 2.1 Features and their challenges in Hindi utterances for performing SLU/DST.

Features	**Example**	**Description of the DST challenge related to a slot**
Morphological features	"मैं शहर के केंद्रीय भाग में बंगाली खाने को खोज रहा हूँ।" (I am looking for Bengali food in the central part of the city.)	Area : "केंद्रीय"(central) is a morphological variant of the word "केंद्र"(centre).
CodeMix features	"मुझे वेजिटेरियन चाहिए । क्या मुझे पता और पोस्ट कोड मिल सकता है? " (I want a vegetarian. Can I have an address and post code?)	Food : The word "वेजिटेरियन" (vegetarian) is an english word that should get mapped to word "शाकाहारी" in Hindi.
Lexical Variations	"मुझे मध्यम रेंज में शहर के ऊपरी भाग में कुछ चाहिए।" (I need something of middle range in the upper part of the city.)	Area : The word "ऊपरी" (upper) is the lexical variant of "उत्तर" (north).
Echo-Words	"क्या आप किसी ठीक ठाक मूल्य के रेस्टॉरेंट का पता तथा फ़ोन नंबर दे सकते हैं ? " (Can you give the address and phone number of a reasonably priced restaurant?)	Price : The phrase "ठीक ठाक" (theek thaak) is an echo word and it meaning is closer to "मध्यम" (moderate).
Hidden Information	"मुझे एक रेस्तरां चाहिए ध्यान रहे मैं गरीब हूँ।" (I want a restaurant in the north, remember that I am a poor.)	Price : The word "मैं गरीब हूँ" (I am poor) indirectly indicates the low-cost restaurant requirement.
Don't care values	"मेरी कोई खास पसंद नहीं है।" (I don't have anything special.)	The phrase ("खास पसंद नहीं") gives an indication of "don't care", associated to any *informable* slot.
Newer slot values	"पूर्व के किसी सस्ते रेस्टोरेंट का फ़ोन नंबर और पोस्टकोड बताइये जो आलू भरे पराठे देता हो? " (Give the phone number and postcode of a cheap restaurant that serves potato filled parathas?)	Food: In the examples, "आलू भरे पराठे" is a newer food slot that did not appear in training dialogues.

network-based models with additional linguistic capability, e.g. BERT [102], have been investigated recently and achieved state-of-the-art results [103–108].

Following the data-driven approach, a new system is developed in the current work by employing a new training corpus in the Hindi language. However, any generic approach may not perform well in situations where context-sensitive information is captured. Several dialogue corpora have been released in the past (Briefly discussed in Chapter 3.2), but no one has explored an Indic language. Depending on whether it is labelled using the structured annotated scheme, these corpora can be categorised into two classes: corpora labelled with structured annotations [54, 93, 109–113, 101]; corpora without semantic labels but having a specific goal during each conversation [114–116]. However, they are limited in terms of proper annotations or built, focusing primarily on the English language.

In addition, Hindi is a morphologically rich language containing lots of lexical/morphological ambiguities, such as inflectional morphology [42], code-mix ambiguities [46, 49], lexical variations [45], echo-words [48] and hidden information [47]. Table 2.1 presents the list of features and their corresponding challenges. There is usually a limit of 8-9 inflected word forms of nouns in English, but in Hindi, it is more than 40 [43, 44]. It becomes a key challenge for DST models to detect the DAs and keep the dialogue state updated appropriately. For the empirical analysis of *language-specific* and *language-independent* challenges in dialogue state tracking, this

thesis proposes a dialogue corpus HDRS to train DST models in a *new language Hindi* with better annotations and high language-variability with significant corpus size, described more detail in Chapter 3.

2.4 Modelling Dialogue Management

Once the dialogue state has been inferred from the DST component, the system's *dialogue manager* must choose an appropriate response. Working as the first component during the system response generation, as shown in the bottom right of Figure 2.1, the dialogue manager component not only controls the flow of the interaction between the system and the user, but it is also a central part responsible for the overall quality of the user experience. Its behaviour is modelled by a dialogue policy, whose job is to map the belief states (dialogue states) to system actions.

The most straightforward approach is to use hand-crafted methods to maintain and control the dialogue interaction. In such cases, the dialogue systems usually utilise the expert knowledge and manage the overall dialogue flow in a flowchart-based system [117, 118], form-filling systems [119] and logical inference and planning on tree-structured knowledge [120]. However, none of these approaches suggests a well-established and systematic way of learning and requires a large amount of human effort and needs significant effort in further modification if there is any update in the domain's ontology and structure. Furthermore, the uncertainty induced by noisy ASR and SLU/DST modules often lead to erroneous DST results, bringing the system to an incorrect state and causing it to take incorrect actions and potentially generate an undesired system response.

Therefore, it has been suggested that statistical methods [121–124] can resolve the limitations of the rule-based approaches. Probabilistic methods in dialogue management have the capability to model uncertainty in the system's belief state and map it into a distribution over sets of system actions. It helps the system to be more robust towards various noisy conditions. A dialogue policy can be implemented as a classification task that trains on dialogue corpus data. Such dialogue managers are highly portable and extendable across different domains.

However, supervised learning of dialogue management faces severe sparsity issues as the dialogue domains are usually exponential in the number of distinct instances they can generate. Even a very large dialogue corpus would represent only a tiny fraction of the total set of plausible dialogues. Hence, a significant amount of abstraction is required to limit the space of dialogue behaviour that can be learnt. However, leveraging such supervised behaviour does not guarantee that it would lead to a successful dialogue [125].

An alternative is to use Reinforcement Learning (RL), where the dialogue interaction is considered as a long-term planning task, with optimising its action selection policy with respect to an objective measure [126]. Unlike the supervised learning models, where the dialogue manager's behaviour is restricted to the type of corpus used, a dialogue manager using RL can explore all possible behaviour.

We model the dialogue policy with RL based approaches where the system's goal is to choose a sequence of system responses (actions) given the observed belief state achieving the maximum total reward, whereby the success of the dialogue mainly determines the reward. Casting it as a Markov Decision Process (MDP), the dialogue policy can learn the action-selection model directly from the interactions [126, 127]. However, learning dialogue policy based on a point estimate of the dialogue state is not ideal due to the erroneous ASR and SLU/DST components as MDP can model only a single hypothesis. Therefore, the Partially Observable Markov Decision Process (POMDP) [39, 128] is used to build the dialogue policy, which considers the multiple hypotheses as a belief state (distribution over all dialogue states), hence offers a more robust and well-founded framework for statistical dialogue modelling.

The details of applying the MDP and POMDP based RL models to a dialogue policy is briefly discussed in Chapter 4 in association with the experiments on recent state-of-the-art models, i.e. GP-SARSA [124], DQN [129] and A2C [130]. We also show that our version of Advantage Actor-Critic with Experience Replay (A2CER) achieves better performance than the current state-of-the-art NN-based policy learning methods.

2.5 Challenges in Natural Language Dialogue Generation

Obtaining the dialogue act from the dialogue manager, the Natural Language Dialogue Generation (NLDG) module transforms this abstract semantics notation (system dialogue act) back into a text representation. For example, the dialogue act:

`request(food)`

can be transformed to:

"आप किस प्रकार का भोजन खाना चाहेंगे?"

(What kind of food would you like to eat?)

Initially, most NLDG systems were based on *rule-based* approaches [131, 28] or a hybrid of handcrafted and statistical methods [132, 133], which have been widely utilised for their simplicity, robustness and high accuracy in limited domains. One such example of a hybrid rule-based NLDG model is *HALogen*[3], implemented by Langkild et al., which performs reranking on handcrafted candidates using an n-gram Language Model (LM) [134]. The major issues with such systems are the lack of language variability in the output and their scalability to large domains [135].

On the other hand, corpus-based NLDG systems aim to learn the generation rules from a set of data. In 2000, a class-based n-gram LM generator, a type of *word-based generator*, was proposed to generate sentences stochastically for a task-oriented dialogue system [27]. However, inherently it has a very high computation cost, and it is indefinite about covering all possible semantics in the outputs. Hence later, the word-based generators

[3] HALogen is a successor to Nitrogen [132].

were replaced by *phrase-based generators*, which had not only reduced the computation cost but also generated linguistically varied utterances [136, 29]. However, the phrase-based generators are restricted to semantically-aligned corpora, which are tedious and expensive to collect.

More recently, researchers have used methods that do not require aligned data and perform *end-to-end* training to get sentence planning and surface realisation done in one go [137]. For achieving the naturalness, variation and scalability on unaligned corpora, they incorporated the deep-learning models. The successful approaches use the RNN-based models to train the *encoder-decoder* on a corpus of paired DAs and corresponding utterances [32, 33]. Wen et al. proposed various Recurrent Neural Network Language Generation (RNNLG) models, i.e. Attention-Based Encoder-Decoder (ENC-DEC), Heuristically-gated LSTM (H-LSTM) and Semantically Controlled LSTM (SC-LSTM), which are also shown to be effective for the NLDG module in task-oriented dialogue systems [12, 138]. Although the deep-learning methods are supposed to learn a high level of semantics, but they require a large amount of data for even a small task-oriented system.

However, none of the previous works has explored natural language dialogue generation on a Hindi SDS. The language divergences between Hindi and English shows that it is more challenging to perform natural language dialogue generation in Hindi [139, 140]. Though Hindi and English belong to the Indo-European language family, they have differences in terms of sentence structure. Hindi has an SOV (Here, S=Subject, O=Object and V=Verb) structure for sentences, while English follows the SVO order [141]. Assuming S_m as a subject modifier, O_m as object modifier, V_m as expected verb post-modifiers and C_m as the optional verb post-modifiers, an example is given as:

(Hi) a. S C_m O_m O V

b. [मैं]$_S$ [शहर के दक्षिणी हिस्से में]$_{C_m}$ [बंगाली]$_{O_m}$ [खाना]$_O$ [ढूंढ रहा हूँ]$_V$।

(En) a. S V O_m O C_m

b. [I]$_S$ [am searching]$_V$ [for Bengali]$_{O_m}$ [food]$_O$ [in the southern part of the city]$_{C_m}$.

It is observed that the case markers, i.e. में (mein), से (se), को (ko), का (kaa) etc. are postpositioned and are strongly bound to nouns. This is why Hindi is a relatively free-word-order language. On the other hand, English uses prepositioned phrases as qualifiers or complements, making the order of words fixed. The free word ordering, however, makes the analysis of Hindi utterances challenging. The task of distinguishing clauses and phrases from the subject and object get difficult. In addition, the morphological variations are richer in Hindi than in English, as previously mentioned in Section 2.3.2.

In our work, we have explored several state-of-the-art RNNLG-based models and compared them with the benchmark models, i.e. Hand-Crafted (HDC), K-Nearest Neighbors (KNN) model and, n-gram model with discussing their performances on language-related (Hindi) challenges. All the models are experimented on our own Hindi dataset, collected on the restaurant domain.

2.6 Speech Synthesis & Quality Evaluation

2.6.1 Speech Synthesis Models

At the last step in the SDS pipeline, the speech synthesis component converts the chosen text or the symbolic linguistic representation into a speech waveform. For the current study, we aim to cover leading TTS technologies as used in research as well as state-of-the-art commercial systems. Both TTS datasets are used to build four types of unmodified "off-the-shelf" TTS systems: Unit selection speech synthesis (USS), Hidden Markov Model speech synthesis (HMM), Clustergen speech synthesis (CLU) and Deep Neural Network-based speech synthesis (DNN).

The USS is fundamentally a cluster-based technique that combines the units of similar type (e.g. phones, diphones, syllables etc.) are clustered based on their acoustic differences [34]. The clusters are then indexed based on high-level features such as phonetic and prosodic context. But its use in the embedded systems gets affected by their computational processing power and memory footprint. It is necessary to find a favourable compromise between the size of the speech corpus and the computational complexity of the unit-selection method [142]. In this thesis, MaryTTS[4], an open-source tool, is used to build the USS models on both CMU and IITM datasets [143]. Phone and Half-phone based contextual feature weights are considered as a base of units used for training and selection [144].

In contrast, the parametric synthesis based TTS systems are specific counterparts to the issues mentioned above; Hidden Markov Model (HMM) based model is one of them [145]. The HMM-based TTS system works in two-phase; the first is to extract temporal parameters, e.g. spectral (e.g., Mel-cepstral coefficients) and excitation features (e.g., log F0 and its dynamic features) from the speech database and then model them. We have built two separate models for CMU and IITM datasets. The second phase generates a sequence of desired speech parameters through trained models for a given word sequence to be synthesised. The parameters sequence with the maximum output probability is considered for forming the final sound wave [146]. It has several advantages over USS and disadvantages too. Many advantages are related to its flexibility in handling different variations efficiently due to its parametric-statistical nature, enabling it to transform (adapting) voice characteristics, speaking styles, and emotions. It has some major drawbacks, too, over USS as the output voice is not that natural.

The CLU is a closer sibling of parametric TTS models, but it also has some characteristics of USS's as well, like selecting a unit from a set (cluster of similar units) rather than based on the contextual cues [147]. As a general TTS system, CLU also requires a set of pairwise spoken utterances and text transcriptions. It models the acoustic features (e.g. MFCC, F0) in the form of CLUNIT extracted from each segment of the speech wave,

[4]The MARY Text-to-Speech System (MaryTTS) http://mary.dfki.de/

using the Classification And Regression Tree (CART) [148]. Additionally, a duration CART tree is also built to model durational variation. The synthesis process starts with converting the input text into a phone string. Each phone links further to three sub-phonetic HMM-states[5]. These sub-phonetic units are going to be processed by respective duration-CART and HMM-state CART combinely to generate averaged track coefficients which are used to synthesise speech using Mel Log Spectrum Approximation (MLSA) filter [149].

Recently, several Deep Neural Network-based speech synthesis (DNN) based autoregressive models for TTS have been proposed, such as WaveNet [36], Deep-Voice 1, 2 & 3 [150–152], Tacotron-1 [37] and Tacotron-2 [38] etc. We use Tacotron-2[6] to build a DNN based TTS, an end-to-end TTS system that is better at handling the missing spectral information. The model first predicts Mel-scale spectrograms from the character embeddings of Hindi letters through a sequence-to-sequence recurrent network, followed by a separate autoregressive model (WaveNet[7]) to turn it into a waveform. The intermediate features (80-dimensional audio spectrogram) computed on 12.5-millisecond frames are not only capable of capturing the pronunciation of the words but also various nuances of human speech, i.e. volume, intonation and tempo.

2.6.2 Quality Evaluation

Evaluation of synthesised speech is considered to be an important but challenging area due to low understanding and exploration of quality aspects of synthetic speech. However, the *speech quality* term is mostly addressed and explored in the area of speech coding or speech enhancement [153, 154]. The assessment of modified speech is usually measured in terms of change in speech quality before and after the modification. On the other hand, synthesised speech requires the assessment algorithms to be far beyond the signal-comparison, which decides the overall adequacy of a synthesis model [155].

Speech quality is commonly a quantification of perception and assessment process by a subject through comparing *perceptual features* as per individual expectations, appropriate requirements and social demand [156]. Subjective (listening) test thus become an important evaluation criterion to measure the psycho-physical property of a synthesised speech. Intelligibility tests, Diagnostic/Modified Rhyme Test (DRT/MRT), the Cluster Identification Test (CLID), Standard Segmental Test (SST) and Semantically Unpredictable Sentences (SUS) tests are a few renowned subjective assessment methods that commonly consider a large group of subjects to deliver appropriate ratings on certain pre-defined criteria [155, 157, 158]. In addition, various *prosody assessment* criteria have been proposed, which are especially crucial for the naturalness of the synthesis [159, 160]. International Telecommunication Union (ITU) has standardised the test protocols for overall quality assessment in the area of speech coding [161, 162].

[5]Edinburgh Speech Tools Library http://www.cstr.ed.ac.uk/projects/speech_tools/
[6]Tacotron-2: https://github.com/Rayhane-mamah/Tacotron-2
[7]WaveNet vocoder: https://github.com/r9y9/wavenet_vocoder

Due to inherent inter- as well as intra-user rating inconsistencies, the listening tests are hard to be proven entirely. Based on the degree of being accustomed to synthesised speech, the perceptual-quality ratings vary significantly. Additionally, understanding the exact relationship between the acoustic characteristics of synthetic speech and listeners' response is not well understood [163]. Regardless of the existing limitation, the listening tests are considered to be the only standard subjective evaluation of the synthesised speech. However, its use in the assessment is restricted as it requires significant human and financial resources. It has set the stage for the use of *instrumental* (statistical) models for quality prediction in recent years [164]. These models are built on the basis of listening-test ratings as well as *acoustic-measurable* properties of the speech and mimic its perception-quality. They are the statistical models which derive averaged auditory ratings as a multivariate function that maps input variables (acoustic features) to an output variable (auditory rating).

The instrumental quality evaluation of the synthesised speech is studied in two research areas (i) the detection and/or quantification of *discontinuities* and (ii) estimation of overall quality. A number of studies have been presented on the first issue of identifying the discontinuous joints in speech [165–168]. It has been pointed out that the focus on such restricted perceptual features and its correlation with the corresponding acoustic properties do not deliver promising results. This rather compact prediction model to a single quality element seems to be inadequate to capture the strong cognitive interactions of perceptual features in a synthesised speech. Alternatively, the development of an *integral quality* prediction model has been proposed as the second line of instrumental quality evaluation. Deriving the reference patterns from the natural speech signal, pattern-recognition approaches have been used to evaluate synthetic speech based on the basic features, i.e. MFCC [169, 170].

An explicit comparison of natural speech directly with the synthesised speech signal has also been investigated in the area of conventional objective evaluation. The comparison method usually involves time-alignment and perceptual modelling of *auditorily-relevant* features, e.g., Perceptual Evaluation of Speech Quality (PESQ) [171] as well as other distortion measure criteria, e.g., Mel-Cepstral Distortion (MCD), Linear Predictive Coding Coefficients (LPCC) Spectral Distortion. However, the differences in the results shown by several studies are not very supportive of its use in evaluating the synthesised speech [172–175]. The imperfect-alignment and differences in speaker-characteristics are found to be major obstacles in such signal-based comparison.

Further, it has also been proved that the intrusive methods could be helpful in explicit tuning based on acoustic feature adaptation. Valentini-Botinhao et al. had explored the tuning of an HMM-based TTS in order to investigate intelligibility on various noisy conditions [176]. Möller et al. used the ITU-T Recommendation P.563 parameters to train general regression models for predicting the quality of synthesised speech [177]. Later, based on these studies, several instrumental models have been thoroughly explored in the quality prediction of

synthesised speech [163, 178]. However, they all require golden (natural) speech files in order to estimate the robustness of an assessment method.

Recently, various non-intrusive assessment methods have been proposed, which utilises neural-network models to perform speech quality evaluation. Tang et al. have evaluated mandarin TTS using the LSTM model on MFCC and P.563 internal feature parameters by predicting the Mean Opinion Score (MOS) of naturalness metric [179]. In the area of Voice Conversion (VC), several end-to-end speech objective assessment models have been proposed: Quality-Net [180], MOSNet [181], Deep-MOS predictor [182]. They all are explored on the evaluation of the speech being synthetically transformed, e.g. corrupting or enhancing the speech signal [180] or speaker identity conversion through data-driven VC techniques [183–185].

In summary, all mentioned studies are concluding towards the following points:

(a) Quality assessment of the synthesised speech needs to be explored acoustically in a *holistic manner*. Both perceptual, as well as integral quality features, are required to be investigated in order to understand how the quality is constructed in a listener's mind.

(b) A simple comparison of synthetic speech with natural speech at a physical level is not reliable to explain the perceptual quality. Hence, unlike in coding & transmission, synthetic speech can not be seen as a *degraded* variant of natural speech but to be considered as the speech of its own class.

(c) Generally, the approaches that use segmental (internal) features (e.g. concatenation points, transition cost, joint cost, etc.) for quality assessment seem to be less potent than the non-intrusive approaches which explore supra-segmental features from the acoustic signal.

2.7 Dialogue Agent & Web Interface

Designing speech interface based conversational systems has been a focus of research for many years. A typical SDS is based on a modular architecture consisting of input processing modules, i.e. speech recognition & language understanding, dialogue management modules, i.e. belief tracking, policy, and output processing modules, i.e. language generation and speech synthesise, as shown in Figure 2.1. In a statistical SDS, all modules are statistical models learned from task specialised corpus. Some of the recent work of various dialogue system components is available in [125, 14, 186–188, 2, 189, 12, 190, 109, 100].

The dialogue agent should fully implement and follow the modular architecture of a spoken dialogue system comprised of all the necessary components, i.e. SLU, DST, DM and NLDG. We incorporated and adapted the multi-domain statistical dialogue System toolkit *PyDial-Toolkit* [191] to build our dialogue agent "SILPAssistant".

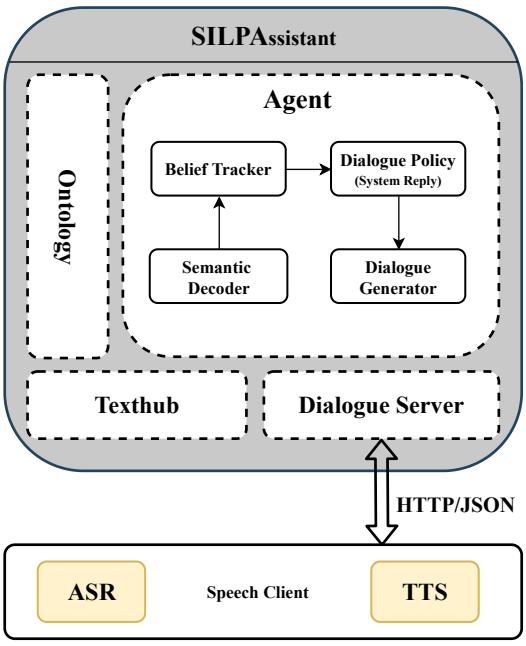


Figure 2.4 The general architecture of SILPA. The Agent resides at the core and, the interfaces Texthub, Dialogue Sever provide the link to the environment.

The general architecture of the dialogue system with a speech interface is shown in Figure 2.4. The **Agent** is the main component responsible for the dialogue interaction. Hence, its internal structure is similar to the pipelined SDS architecture presented in Figure 2.1. It consists of dialogue system modules of semantic decoder, belief tracker, policy and language generator.

Figure 2.5 SILPA: Web-based interface to a dialogue agent.

The Agent can communicate to the user in both texts as well as speech. For the text-based interaction, **Texthub** utility is provided, which simply connects the Agent to a terminal. To enable speech-based dialogue, the **Dialogue-Server** works as an interface between the Agent and the Speech-Client. The Speech-Client pro-

vides the facility of Automatic Speech Recognition (ASR) and Text-To-Speech (TTS) to interact with the user in speech mode. It connects to the Dialogue-Server via HTTP/HTTPS exchanging JSON messages. Along with the Agent and Interface components, the dialogue system also consists of an **Ontology** that stores the application-domain specification as well as access to the back-end database, e.g. set of restaurants with corresponding properties.

In order to provide real-time interaction, we developed a web-based (portrait is shown in Figure 2.5) as well as a mobile-based application. It has the capability to establish and maintain a conversation with a pool of real users to a set of virtual dialogue agents. Here the virtual dialogue agents mirror the same back-end **Dialogue Agent** built on the aforementioned toolkit. It uses the Google Chrome Speech ASR API to transform the user's speech into text and the Google Chrome TTS API to convert SILPA's text output into Hindi speech.

2.8 Summary

The chapter has presented an overview and outlined the research-gap for designing each module in the domain of task-oriented spoken dialogue systems. Before providing the overview of each modular component, we first highlighted characteristics of a typical modular SDS, i.e. task-oriented systems, domain ontologies, dialogue act, an analogy of the uncertainties in dialogue with computer network scenario in the perspective of Hindi language. Based on the modular architecture, the overall research in the area of statistical spoken dialogue systems is focussed on constructing all the system components as statistical models with parameters learned directly from the data by resolving various language-specific and language-independent challenges. After discussing the roles of each component with existing state-of-the-art, the chapter has presented a framework named 'SILPA', which establish the speech-based communication between the agent and the user.

The next chapter will provide a detailed description of an introduced HDRS corpus and the comparison of several state-of-the-art SLU/DST models on it.

Chapter 3

HDRS: Language Understanding & State Tracking

3.1 Introduction

In this chapter, we raise the key research questions that underlie the SLU and DST module in building a Hindi dialogue system for restaurant domain:
The first challenge is in the SLU module [192],[89],[193] where the system should automatically identify the Dialogue-Act (DA) of the user query (i.e. User-Act) [194]. It includes finding the User-Act type and then grabbing the slot-values corresponding to food, price and area in the dialogue. Let us consider the following user query:

User :	"मुझे शहर के उत्तर भाग में एक उचित मूल्य वाला रेस्तरां चाहिए।" (I want a restaurant in the northern part of the city with a reasonable price.)

The query belongs to DA Type="inform" and provides information about the person's preference regarding "area" and "price range". These are called 'Slots'. The values are the entities associated with these slots, i.e. "area"="उत्तर" (north) and "price range"="मध्यम" (moderate). Hence the corresponding DA of the query is presented as: (i.e. `inform(area="उत्तर", price range="मध्यम"))`

Hindi is a morphologically rich language containing lots of lexical variations. Some of the lexical variations of the above user utterance in the corpus are:

1. "कृपया कर मुझे शहर के ऊपरी भाग में एक रेस्तराँ बताइये जिसकी कीमत बीच की हो।"
   (Please tell me a restaurant in the upper part of the city that costs in mid range.)

2. "क्या आपके पास कोई ऐसे भोजनालय की जानकारी है जो ना ज्यादा महंगा और ना ही ज्यादा सस्ता और शहर के उत्तरी भाग में हो? "
   (Do you have information about any restaurant that is neither more expensive nor more cheap and in the northern part of the city?)

3. "मुझे शहर के ऊपरी भाग में एक ठीक ठाक मूल्य वाला रेस्टोरेंट चाहिए।"
   (I would like a reasonably priced restaurant in the upper part of the city.)

In all the statements, the user conveys similar intentions, and hence they should be mapped to the same DA. The second challenge is in the DST module, where the system needs to keep its dialogue state updated using the dialogue history [195–198].

In a traditional dialogue system pipeline, Spoken Language Understanding (SLU) and Dialogue State Tracking (DST) were separate. These trackers are prone to accumulating errors received from the SLU module as it sometimes propagates unnecessary dialogue context. Subsequently, research on belief trackers gets focussed on conceptualising SLU and DST as a single module [97–99]. To achieve better generalisation, these trackers rely on hand-crafted *semantic-dictionaries* and *delexicalisation*. In papers [100, 24], Convolutional Neural Network (CNN) based representation learning had been applied to learn relevant features from semantically-induced word embeddings, e.g. PARAGRAM-SL999[1], GloVe etc. to predict each state stochastically without relying on any hand-crafted features.

Global-Locally self-Attentive Dialogue State Tracker (GLAD) was proposed as an improvement by employing an encoder with two separate modules: one for sharing parameters between the slots through a *global-module* and other for learning slot-specific features through a *local-module* [103]. It helps in generalising the rare slot-value pairs with few training examples. GLAD-DST is further extended by a different encoder Globally-Conditioned Encoder (GCE) that avoid using inefficient recurrent and self-attentive layers in the encoder [104]. Although GCE had simplified the GLAD-DST model, still requires a separate encoder to extract features in each turn from the utterance, system action and the candidate slot-value pairs which causes high *time complexity* and significant latency in the prediction. To mitigate this [105] proposed new augmentation of Global encoder and Slot-ATtentive decoder (GSAT) in the Belief-Tracker architecture which efficiently handles the latency time with approximately more than 20-times faster than the previous state-of-art DST models.

After the introduction of the contextual semantic word vectors such as BERT [102], DST models such as BERT-DST [106] , Simple-BERT DST [107] and Slot-Utterance Matching for universal and scalable Belief Tracking (SUMBT) [108] were introduced. These model architectures not only share the parameters among the slot values but also provide the capability to extend new slot-values in the ontology during the testing phase.

The contribution of our work lies in the following aspects:

[1]Generated by injecting similarity constraints from the Paraphrase Database into GloVe [199]

1. To release a Hindi dialogue corpus containing a large number of labelled dialogues on the restaurant domain.
2. To provide the details of features, collection process and statistical analysis of the proposed corpus.
3. To show the performance over baseline models for SLU and DST tasks.
4. To compare the performance of state-of-the-art DST models over the released corpus.

The chapter is organised as follows: current Section 3.1 presents introduction of the chapter discussing challenges of SLU and DST task and related work. Section 3.2 of the current chapter presents related work and compares various dialogue corpora categories. Section 3.3 describes the features, collection process and statistical analysis of the proposed corpus. Section 3.4 presents a brief description of baseline and state-of-the-art DST models, whereas Section 3.5 gives details about the experiment performed on the released corpus. Section 3.6 discusses the result and analysis. Section 3.7 concludes the chapter.

3.2 Related Work

In a data-driven approach, a new system can be developed by employing a new training corpus. However, any generic approach may not perform well in situations where context-sensitive information is captured. Several dialogue corpora have been released in the past. Depending on whether it is labelled using the structured annotated scheme, these corpora can be categorised into two classes: corpora labelled with structured annotations [54, 93, 109–113, 101]; corpora without semantic labels but having a specific goal during each conversation [114–116]. However, they are limited in terms of proper annotations or built, focusing primarily on the English language. The proposed corpus has been created to train a DST in a *new language Hindi* with better annotations and high language-variability with significant corpus-size.

Based on the way of collecting the conversations, existing datasets can be grouped into three major categories: Machine-to-Machine (M2M), Human-to-Machine (H2M) and Human-to-Human (H2H) conversations [200]. M2M paradigm has the capability to generate an infinite number of dialogue templates with a simulated user. These templates can then be transformed in natural-language with the help of either pre-defined rules [201] or crowdsourcing M2M [112], Schema-Guided Dialogue (SGD) [113]. However, such a paradigm covers all possible dialogue scenarios within a certain domain. It has some serious limitations. As all the conversations are engineered-up by the users and system bots, the system could easily hang on any unseen event, e.g., unforeseen flows, misunderstandings or repetitions [202]. Creating a good user-simulator is itself a very hard task to make it right at the beginning.

The idea of H2M paradigm is to launch an initial system that interacts with the real users. One such system is the Let's Go Bus Information System [203] that leads to building similar system in the initial Dialogue State

Table 3.1 Comparison of various datasets for task-oriented dialogue systems. (Hi=Hindi, En=English, It=Italian, De=German and Cn=Chinese)

Name	Type	Language	Avg # of Turns	Total # of Dialogues	Description
ATIS Pilot Corpus [54]	H2H (spoken)	En	25.4	41	First H2H spoken dialogue corpus on air travel planning and booking domain.
CMU Comm. Corpus [204]	H2M (spoken)	En	11.67	15,481	An H2M spoken dialogue corpus on travel planning and booking domain.
DSTC-1 [93]	H2M (spoken)	En	13.56	15,000	First DST challenge of proposing H2M dialogue corpus on bus-ride information domain.
DSTC-2 [94]	H2M (spoken)	En	7.88	3,000	DSTC on restaurant booking system with dynamic user-goal and richer dialogue state challenge.
DSTC-3 [56]	H2M (spoken)	En	8.27	2,275	A dialogue corpus on tourists domain with the study handling new slot-value during the testing.
DSTC-4&5 [95, 96]	H2H (spoken)	En, Cn	32	35	First DST challenge to provide H2H-type dialogue corpus collected on tourist domain.
WOZ 2.0 [109, 100]	H2H (typed)	En, It, De	4	1,200	A dialogue corpus of conversations on restaurant domain.
Frames [110]	H2H (typed)	En	14.60	1,369	A goal-driven dialogue corpus on the travel domain provides the challenge of complex decision-making behaviour.
KVRET [111]	H2H (typed)	En	5.25	2,425	A multi-domain dataset designed to investigate conversation interface with an explicit knowledge-base.
M2M Corpus [112]	M2M (typed)	En	9.86	1,500	Machine-generated corpus with customised diversity and coverage on movie and restaurant domain.
SGD [113]	M2M (typed)	En	20.44	16,142	Schema-Guided Dialogue (SGD) dataset, containing over 16k multi-domain conversations spanning 16 domains.
MultiWOZ [101]	H2H (typed)	En	13.46	8,438	A large-scale multi-domain dialogue corpus.
HDRS (proposed)	**H2H (typed)**	**Hi**	**4.12**	**1,400**	**A Hindi dialogue corpus of conversations collected on restaurant domain (in Allahabad - a city in India).**

Tracking Challenges (DSTC), e.g. DSTC-1 [93], DSTC-2 [94] and DSTC-3 [56]. Provisioning of an initial system (chicken-egg problem) limits this approach to be used in the improvement of the existing systems. Due to the initial system's limited capability, the users adapt to simpler utterances instead of expressing natural sentences.

H2H is the most intuitive approach to build a natural conversational system trained on a large human-human corpus. Based on this fact, several large-scale corpora have been released in the past, such as Twitter Conversations [114], Ubuntu Technical-Support Dialogue Corpus [115], Reddit Conversations Corpus [205], Persona-Chat [116]. Although these corpora are shown effective in generating interesting responses [206], due to the lack of the explicit goal in the conversation, such corpora are hard to evaluate and struggle to generate consistent and diverse responses [207]. ATIS corpus [54] is one of the earliest datasets, collected in H2H manner to design a spoken dialogue corpus to study both speech and language components in Spoken Language Systems.

Recently, WOZ 2.0 [109, 100], FRAMES [110], KVRET [111], MultiWOZ [101] have shown the usefulness of the WOZ approach in collecting high-quality typed conversations. The typed conversations corpora have an advantage over the spoken corpora on measuring semantic understanding effectively rather than focussing on robustness to ASR errors. In comparison with the H2M, e.g. CMU Comm., DSTC-1,2,3 corpora; H2H corpora such as ATIS, DSTC-4&5, FRAMES, KVRET, WOZ 2.0, MultiWOZ, gave more freedom to the users to use more sophisticated language as the users would quickly adapt to the system's language understanding capability

in the former. WOZ 2.0 [100] was the first to adopt this scenario with the typed-WOZ paradigm. In the area of conversational system in the Hindi language, Sumit et al. [52] have proposed a Hindi dataset suitable only to build a natural language dialogue generation module.

Table 3.1 presents the comparison of various task-oriented dialogue corpora. In the thesis, we aim to build first Hindi dialogue dataset collected in H2H manner using Wizard-of-Oz paradigm which was initially proposed by Kelley et al. [208] as an iterative approach to improve user experiences while designing a dialogue system.

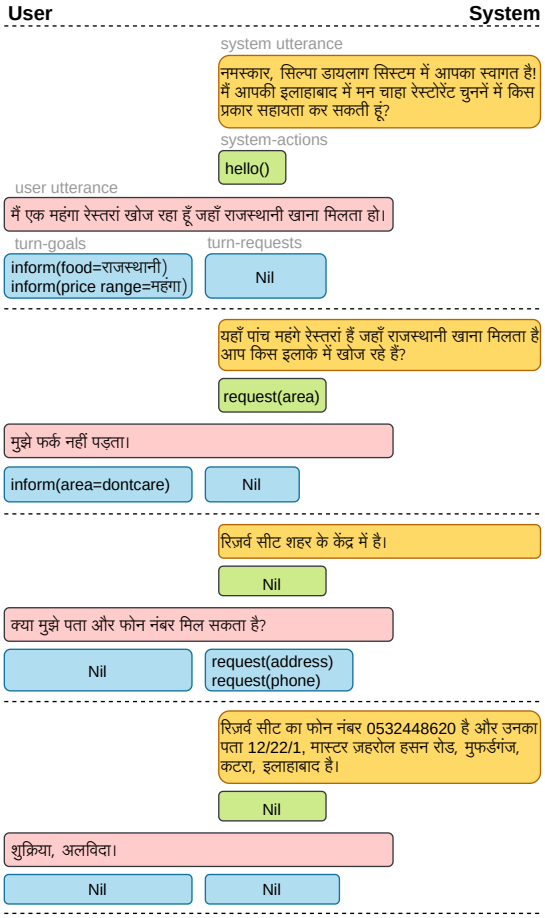


Figure 3.1 A dialogue from the HDRS corpus collected on restaurant domain. Each turn, separated by the dashed lines, contains a *system utterance* (yellow) followed by corresponding *system-actions* (green) as well as *user utterance* (red) comes with the specified *turn-goals* and *turn-requests* (blue). Box with 'Nil' entry depicts the unexpressed entity. Appendix C presents the translation of utterances expressed in the conversation.

3.3 Hindi Dialogue Restaurant Search (HDRS) corpus

The **Hindi Dialogue Restaurant Search (HDRS)** corpus is collected to promote research and development in the field of Hindi dialogue system. In this work, it is used for the evaluation of various SLU/DST models. The corpus revolves around a person whose primary language is Hindi and is searching for a restaurant in Allahabad[2].

[2]https://en.wikipedia.org/wiki/Allahabad

It contains 1.4k handwritten dialogues collected using *Wizard-of-Oz* fashion. To the best of our knowledge, this is the first attempt to release a Hindi dialogue corpus. The corpus is freely available at URL:

https://github.com/skmalviya/HDRS-Corpus

The corpus uses the details of 118 Indian restaurants. The details of most of the restaurants' such as name, phone, address, postcode, were altered. Table A.1 presents the distribution of restaurant database based on the price range and area in Appendix A.

Figure 3.1 presents a sample system and user conversation from the corpus. The dialogues collected are *system-initiated*. A single pair of system and user utterance is referred to as a *turn*. The Dialogue-Acts (DAs) supported by the corpus are specified in the ontology (Table A.2 in Appendix A). The *inform* type DA applies the restrictions (such as `inform(area="उत्तर", price range="मध्यम")`) by the user while searching the restaurant. The *request* type DA are slots used to fulfil the request demands (such as phone, address, postcode, food, area and price range).

The structure of each dialogue in the corpus is as follows:

1. *dialogue_idx:* A unique index for dialogue identification.
2. *dialogue:* It contains a collection of *turns* in a dialogue. Each *turn* is a pair of system and user utterance. The information present in a turn consists of:
   (a) *turn_idx:* A index value to identify a turn in a dialogue uniquely.
   (b) *transcript:* The user utterance in written form.
   (c) *turn_label:* The DA corresponding to the current user utterance.
   (d) *belief_state:* The updated current state of the dialogue. The state summarises the history of the dialogue to provide necessary details to choose the next move by the system [209]. Therefore it maintains the DAs record.
   (e) *system_transcript:* The system utterance in written form. The system utterance in the corpus either conveys fetched information from the database or requests the user for more information to reach the goal.
   (f) *system_act:* The System DA corresponding to a system utterance. It can be either be empty when the system utterance conveys only the information fetched from the database or consists of slot-value pair that it wants to confirm/request. The System DA are helpful in capturing the context of the previous turn.

Hindi is a morphologically rich language containing lots of lexical/morphological ambiguities. It becomes a key challenge for DST models to detect the DAs and keep the dialogue state updated appropriately.

3.3.1 Features of the corpus & their challenges

The Table 3.2 presents the list of HDRS features and their corresponding challenges in DST. These features are elaborated as follows:

Table 3.2 HDRS corpus features and their challenges in the DST.

HDRS Features	Example	Description of the DST challenge related to a slot
Morphological features	"मैं शहर के केंद्रीय भाग में बंगाली खाने को खोज रहा हूँ।" (I am looking for Bengali food in the central part of the city.)	Area : "केंद्रीय"(central) is a morphological variant of the word "केंद्र"(centre).
CodeMix features	"मुझे वेजिटेरियन चाहिए । क्या मुझे पता और पोस्ट कोड मिल सकता है?" (I want a vegetarian. Can I have an address and post code?)	Food : The word "वेजिटेरियन" (vegetarian) is an english word that should get mapped to word "शाकाहारी" in Hindi.
Lexical Variations	"मुझे मध्यम रेंज में शहर के ऊपरी भाग में कुछ चाहिए।" (I need something of middle range in the upper part of the city.)	Area : The word "ऊपरी" (upper) is the lexical variant of "उत्तर" (north).
Echo-Words	"क्या आप किसी ठीक ठाक मूल्य के रेस्टॉरेंट का पता तथा फ़ोन नंबर दे सकते हैं ?" (Can you give the address and phone number of a reasonably priced restaurant?)	Price : The phrase "ठीक ठाक" (theek thaak) is an echo word and it meaning is closer to "मध्यम" (moderate).
Hidden Information	"मुझे एक रेस्तरां चाहिए ध्यान रहे मैं गरीब हूँ।" (I want a restaurant in the north, remember that I am a poor.)	Price : The word "मैं गरीब हूँ" (I am poor) indirectly indicates the low-cost restaurant requirement.
Don't care values	"मेरी कोई खास पसंद नहीं है।" (I don't have anything special.)	The phrase ("खास पसंद नहीं") gives an indication of "don't care", associated to any *informable* slot.
Newer slot values	"पूर्व के किसी सस्ते रेस्टोरेंट का फ़ोन नंबर और पोस्टकोड बताइये जो आलू भरे पराठे देता हो?" (Give the phone number and postcode of a cheap restaurant that serves potato filled parathas?)	Food: In the examples, "आलू भरे पराठे" is a newer food slot that did not appear in training dialogues.

- *Morphological features:* Hindi is very rich in inflectional morphology. There is usually a limit of 8-9 inflected word forms of nouns in English [42], but in Hindi, it is more than 40 [43, 44]. As in the case of Hindi verb, it exhibits grammatical information like gender, tense, number, person etc. through inflectional suffixes. In a sentence from our corpus, e.g. "क्या आप मुझे काकेदा होटल का पता भी दे सकती हैं ?" (Can you also give me the address of Kakeda hotel?), the verb phrase, "दे सकती हैं" (Can you give) provides the information about the gender of the object "आप", which in our case is feminine, but in English, the verb phrase does not maintain this information.

- *Code-mix features:* Code-mixing is the mixing of more languages in the conversation. There are many cases in the corpus where the user had expressed some words from English during the conversation. (Example: "मुझे कम रेंज वाले रेस्तरां की तलाश है।" ("I am looking for low range restaurants.")) here the word "रेंज" (range) is an English word which gives an indication of the cost. Therefore, in the belief state tracking, the word "कम" (less) need to be associated with costing after the resolution of the codemix [46, 49]

- *Lexical Variations:* The way a language is spoken and written gets change from place to place. It leads to the introduction of variations where the meaning of a sentence is same, but the way to express gets change [45].

- *Echo words*: It is prevalent in an informal conversation where a meaningful word ("ठीक/theek" (moderately fine)) is followed by a rhymic non-meaningful word ("ठाक/thaak") that adds a more general sense to it. These words provide the speaker's sense of vagueness into it [48].

- *Hidden Information*: It is prevalent in conversation that the people do not convey each and everything they need; instead, they give an indication which makes it more interesting [47].

- *Don't care slot values*: There are user utterances where the user's reply is somewhat like: "मुझे परवाह नहीं है।" (I do not care). Here it becomes important to look into the context to know about which slot the value is being associated with.

- *Newer slot values outside the Training example*: In the corpus, there are some slot values that are absent in the training set but do exist in the testing set. This feature will help in testing whether the model is capable enough to deal with the dynamic ontologies.

3.3.2 Corpus Collection

The corpus is collected in WOZ fashion [208, 210] using text as a medium during the conversation. As per the WOZ settings [211], the user interacts with the system and is unaware of the fact that an operator controls the system response.

Three pairs of connected systems were prepared for the data collection. For conducting the experiments, two separate rooms were chosen: an outer room and an inner room. For each pair, one system was kept in the outer room and another one in the inner room. In the experiment, three people were chosen as operators (*Wizards*) for the systems in the inner room. The users, sitting in the outer room, were given instructions to interact with the systems in the inner room.

The experiment is performed in the laboratory settings of a national academic institute. To experiment, three experienced members of the laboratory were selected as operators (i.e., Wizards). One hundred fifty students belonging to different states of the country were chosen to participate as users in the experiment.

Before starting the experiment, the users were shown the ontology. They were made aware of *Inform* and *Request* type DAs. Some sample goals were also shown to assist them to understand their goals before starting the conversation. Some of these sample goals were:

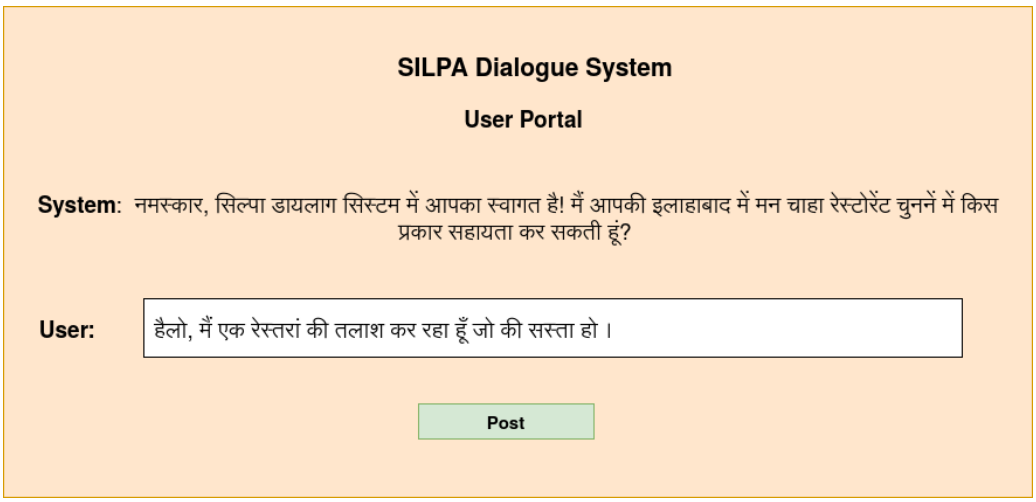


Figure 3.2 User Portal for the WOZ setting. The portal displays all the previous utterances of the dialogue. A text box is provided to fill the user utterance in written form that gets submitted on click of "Post" button.

Sample Goal 1:	आप एक ऐसे रेस्तरां का पता जानना चाहते हैं जो शहर के उत्तर में है और कम कीमत का है।" (You want to know the address of a restaurant which is in the north of the city and is of low price.)
Sample Goal 2:	"आपको एक ठीक ठाक साउथ इंडियन रेस्तरां के फ़ोन नंबर की तलाश है।" (You are looking for the phone number of an average fine South Indian restaurant)

During the experiment, the system at the user's side displays the conversation until that point and a text box for posting the user's response, as shown in Figure 3.2. For writing the Hindi text quickly, the users and operators were told to use Google Input Tool[3]. The users were informed that they need to be creative and should cover many linguistic variations during the conversation.

The operator has access to the database entries for the list of restaurants. The user dialogue was labelled manually by the operator on the fly. These labels are used by the system to fetch the entries from the database, and using it operator frames their response (Figure 3.3). They were also informed to not only fetch the entries from the database and present them in the sentence form to the user but also help the user by giving them suggestions whenever possible, making the conversation engaging. For Example:

[3]Google Input Tool: https://www.google.com/intl/sa/inputtools/try/

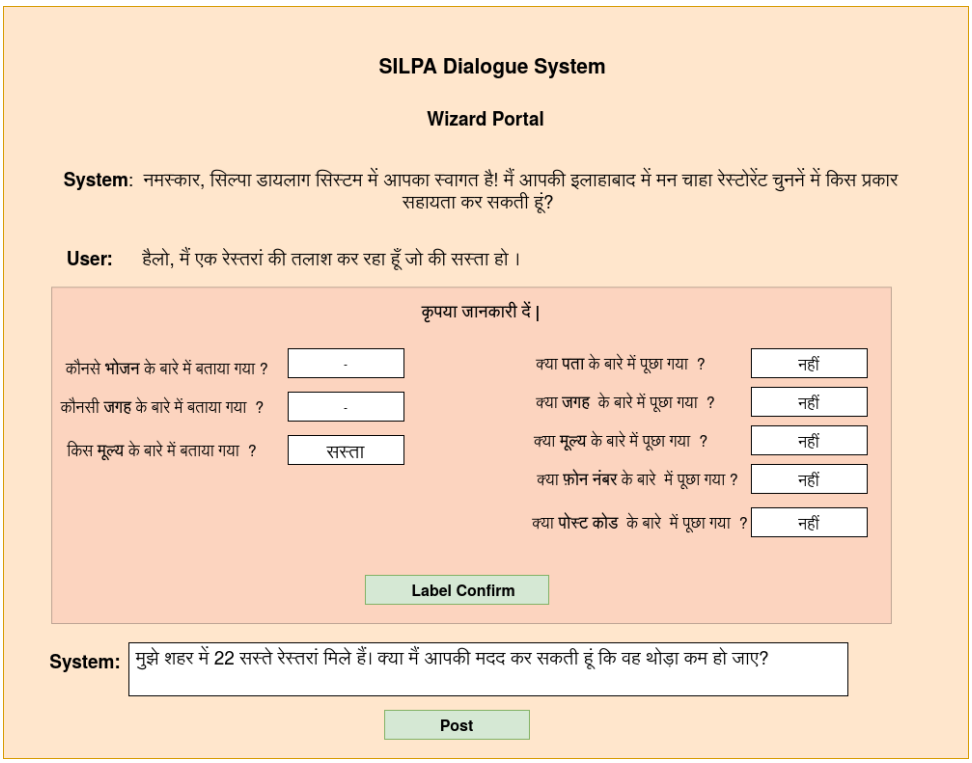


Figure 3.3 Wizard Portal for the WOZ setting. The portal displays all the previous utterances of the dialogue. The text boxes are provided to fill the labels from user utterances manually. "Label Confirm" button saves the labels and use them to fetch records from the database. A text box to fill the system utterance is also provided that gets submitted on click of "Post" button.

User:	"मैं खाने के लिए सस्ते मद्दे कीमत वाले रेस्तरां की तलाश में हूँ।"
	(I am looking for low-priced restaurants to eat.)
System:	"सस्ता मूल्य सीमा में 22 रेस्तरां हैं। आप किस तरह के भोजन में रुचि रखते हैं ?"
	(There are 22 restaurants in the low price range. What kind of food are you interested in?)

The above example presents that the system had fetched 22 restaurants that satisfy the user's needs. At this stage, despite presenting the list of entries directly to the user, the system was made to engage by asking or giving suggestions for choice.

Before beginning the experiment, the inter-annotator agreement was conducted on the operators for all DAs over 450 turns. Fleiss' kappa metric [212] was chosen that gave an average weighted kappa value of 0.980, showing the least discrepancy among the annotators.

At a time, three parallel sessions were conducted. Each day, a target of collecting 10 dialogues per system was set. The experiment was conducted for 48 days which led to around 1.44k dialogues. Finally, the collected data was verified by the team members to check the following:

1. Consistency of the dialogue information with the database.

2. Manual error (e.g. typos) by user and operators while writing Hindi sentences.

3. Irrelevant questions outside the scope of the system.

4. Task completion by the user in the conversation.

The dialogues with the minor problems were corrected, while the major ones were removed. After pruning, the number of remaining dialogues were around 1.44k, which were truncated exactly to 1400 in order to gain ease in the split. Finally, the corpus was split into three parts: the training part containing 800 dialogues, the testing part with 400 dialogues and the validation part with 200 dialogues.

Table 3.3 Statistics of the HDRS corpus for Training, Testing and Validation data.

Properties	**Train**	**Test**	**Valid**
Total #Dialogues	800	400	200
Total #Turns	3288	1645	830
Avg Turns per dialogue	4.11	4.11	4.15
Avg Tokens per user-utterance	8.41	8.53	8.18
Avg Tokens per system-utterance	12.19	12.34	12.00
#Dialogues with goal change	321	161	75

3.3.3 Statistical Corpus Analysis

Table 3.3 presents the statistics of the corpus. The following inferences are drawn:

1. On average, a dialogue in the corpus contains four pairs of the system and user utterances (i.e., the average number of turns per dialogue is 4.11). The distribution of turns per dialogue in the dataset (as shown in Figure 3.4) follows a *normal curve*. The number of turns ranges from two to nine. The similar distribution is seen in all the three splits (i.e. training, validation, testing).

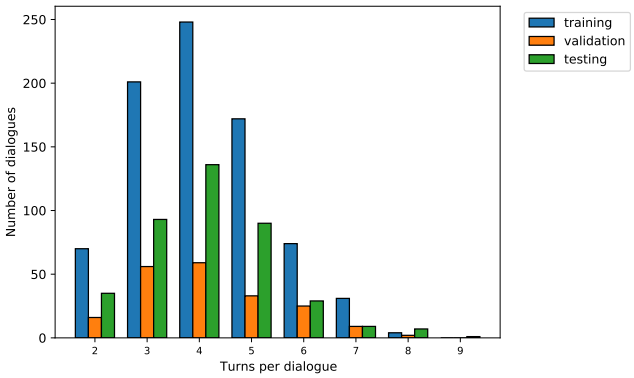


Figure 3.4 Dialogue distribution based on the number of turns per dialogue.

2. The average length of the system utterance is greater than the user utterance. Intuitively it is noticeable because the user is asking queries which are usually short while the system utterance is a response to the query and hence long.

3. *The number of dialogues with goal change* presents the number of dialogues where the user changes her goal [94]. For example, a user starts with 'महंगा' (high price) restaurant in the dialogue but later switches to 'सस्ता' (low price) restaurant by the end.

   It is computed by comparing the belief state of the first utterance and the last utterance of the dialogue corresponding to inform type DA. The following three scenarios are satisfied in the belief state of the system to fulfil the demand presented in the user's first utterance:

   (a) The slot-value corresponding to the first utterance matches with the last utterance.

   (b) The value corresponding to a slot in the first utterance is "don't care", but in the last utterance, it stores value.

   (c) The slot is absent in the first utterance, but in the last utterance, it stores value.

   Similarly, the opposing case of user's first utterance dissatisfaction indicates that the system has failed to satisfy the demand of user mentioned in their first utterance. In these cases, the user has opted to choose alternatives during the conversation.

   As per Table 3.3, there are 40% of the dialogues where the users' goals got changed. Hence the corpus contains sufficient dialogue scenarios that are more natural and challenging for the dialogue state tracking.

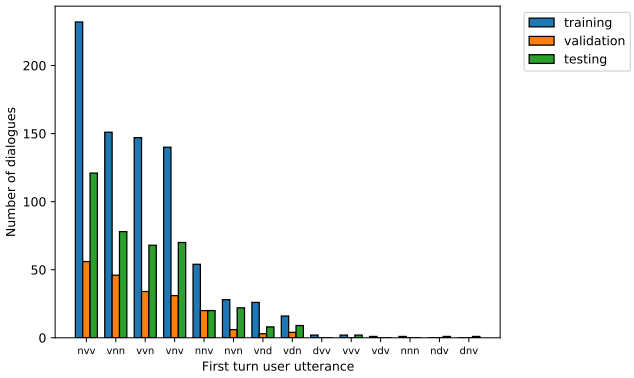


Figure 3.5 Dialogue distribution based on the first turn user utterance. The X-axis plots a tuples (<food_value>,<area_value>,<price range_value>) corresponding to *inform-type* DA. <food_value> can hold any of the three values: 'n'='none' (i.e. user do not mention anything about food), 'd'='dontcare' (i.e. user do not care about any specific food) and 'v'=<some value> (i.e. user mentions the specific food). Similar is the case with other entries in the tuple. Example: nvv indicates `inform{area=<'value'>, price range=<'value'>}`

4. Figure 3.5 presents the slot-value distribution in user's first turn utterance across the corpus. It is observed in the corpus that the most likely way a user starts the conversation is by informing the value of area and

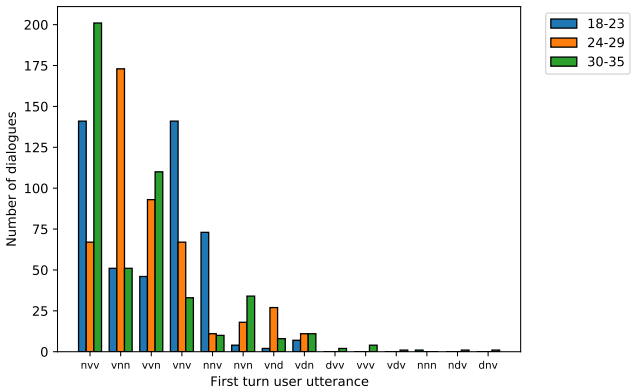


Figure 3.6 Dialogue distribution based on the user's first turn query grouped based on age

price range (i.e. `inform{area=<value>, price range=<value>}`). The next three likely way that the user starts the conversation, are:

- Informing the value of food only (i.e. `inform{food=<value>}`).
- Informing the value of food and area (i.e. `inform{food =<value>, area=<value>}`).
- Informing the value of food and price range (i.e. `inform{food=<value>, price range= <value>}`).

5. The corpus is collected from the people of three different age group. Figure 3.6 presents the slot-value distribution in the first turn grouped by age. It is observed that the users belonging to age-group (18-23) are mostly price-oriented that is they started the conversation by:

   - Informing the value of area and price range (i.e. `inform{area=<value>, price range= <value> }`).
   - Informing the value of food and price range (i.e. `inform{food=<value>, price range=<value>}`).
   - Informing the value price range only (i.e. `inform{price range=<value>}`).

   The users belonging to age-group (24-30) are more food-oriented that is they started the conversation by:

   - Informing the value of food only (`i.e. inform{food=<value>}`).
   - Informing the value of food and area (`i.e. inform{food=<value>, area=<value>}`).

   The users belonging to age-group (31-35) are more area-oriented that is they started the conversation by:

   - Informing the value of area and price range (`i.e. inform{area=<value>, price range=<value>}`.
   - Informing the value of food and area (`i.e. inform{food=<value>, area=<value>}`.

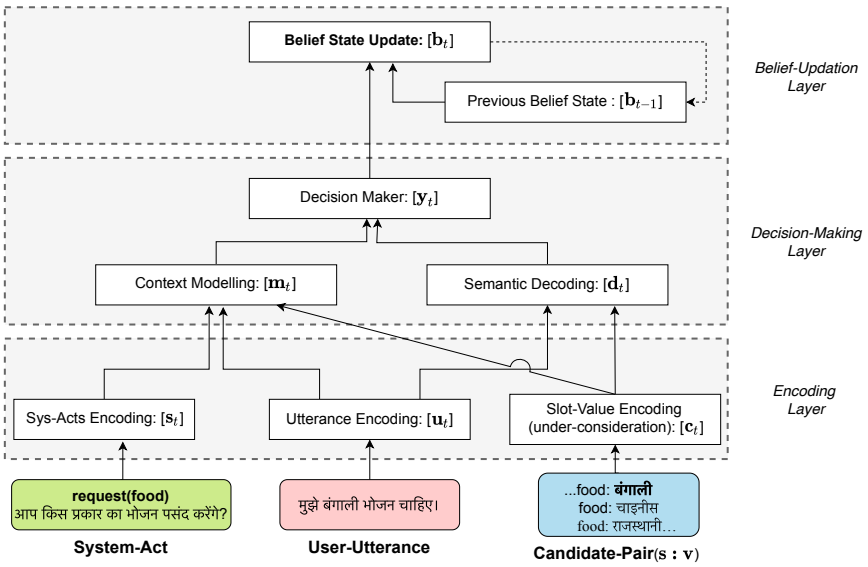


Figure 3.7 Generic architecture of a neural belief tracker.

3.4 Dialogue State Trackers (DST)

In this section, we explain the task of DST followed by the discussion on baseline SLU/DST models and comparison of recently proposed DST architectures: (1) NBT-{CNN/DNN}, (2) GLAD, (3) GCE, (4) GSAT, (5) Simple-BERT DST and (6) SUMBT, that are incorporated on our corpus.

In a task-oriented SDS, a DST generally estimates the distribution over the values V_s for each slot $s \in \mathrm{S}$ based on the *user input* as well as the *dialogue history* up to that turn in the conversation. The task is defined by an *ontology*, which comprises of *informable* ($\mathrm{S_{inf}}$) slots and a set of *requestable* ($\mathrm{S_{req}}$) slots. In this way, the dialogue state in each turn is comprised of *turn-goals* and *turn-requests*. *Turn-goals* are defined by the value for each of the informable slots $s \in \mathrm{S_{inf}}$. The value could either be a value $v \in V_s$, or one of the special values: *dontcare* or *none*. On the other hand, *Turn-requests* are denoted by a subset of requestable slots $\mathrm{S}' \subseteq \mathrm{S_{req}}$, indicate slot-values which user desire to know.

For elucidating the DST task, a dialogue from the corpus is shown in Figure 3.1. The user starts the conversation by specifying a set of turn-goals as `inform(price range=महंगा,food=राजस्थानी)` in the first turn. In the next turn, she adds constraint `inform(area=dontcare)` into the *joint-goal* in response to the system request `request(area)`. When the system conveys a restaurant choice to the user, she requests specific information about that, e.g. `request(address,phone)`.

In Figure 3.7, the generic architecture of a neural-belief-tracker is shown. A DST model consists of three layers in general: *encoding layer*; *decision-making layer* and *belief-updation layer*, in a bottom-up fashion. Encoding layer job is to encode and extract features from the input which are *1) System-Act*[4]- word embeddings for slot-name and its value (e.g. (`food=`गुजराती)) conveyed by the system previously, *2) User Utterance*- a se-

[4]According to [100], our corpus allows system either to request the user to mention the value of a slot (e.g. "आप किस प्रकार का भोजन पसंद करेंगे?") or confirm the a specific slot's value (e.g. "मेरे पास केंद्र में एक हैदराबादी बिरयानी रेस्तरां है। क्या आप उसमें रुचि लेंगे?").

quence of word embeddings of the current user utterance and *3) Candidate-Pair (s:v)*- word embeddings for the slot-value (SV) pair being investigated currently by the model.

Next, the decision-making layer takes the encoded representation of the input and performs the task of *context-modelling* [97] and *semantic-decoding* [195] to decide whether the candidate slot-value has been expressed or not [100]. Based on the decision, belief-updation layer finally updates the *belief-state* stochastically or through a rule-based method.

3.4.1 SLU-Detached Dialogue State Tracking

The SLU-detached DST models take the semantic input from the SLU module and update the current belief state using a rule-based tracker, such as basic and focus trackers [58].

Spoken Language Understanding (SLU)

In the pipeline of a dialogue system, the SLU component's task is to convert the expressed words into a semantic representation as required by the next components. The SLU components must have the capability in dealing with a variety of natural language expressions and able to extract essential details such as slot-value constraints and requested slots from a user utterance. For example, for a user utterance such as "मैं शहर के पूर्व मैं एक महंगा रेस्टोरेंट ढूंढ रहा हूँ। मैं इसका नाम, पता और फ़ोन नंबर चाहता हूँ।" (I am looking for an expensive restaurant in the east of the city. I want its name, address and phone number.) should be mapped to the mentioned constraints `inform(area=पूर्व,price range=महंगा)` and requested slots `request(name, address,phone)`.

We have used the extended *Semantic Tuple Classifier (STC)* based discriminative method as a baseline SLU [195]. It considers the user utterance as a collection of n-gram features on which a multi-class SVM/SGD[5] classifier is trained to detect the DA-type, and a set of binary SVMs/SGDs are trained to detect the expressed slot-value pairs. The SVM classifiers are trained with the linear kernel, while SGDs are probabilistic classifiers based on logistic regression.

At each turn, the input is constructed by combining the features of both user utterance U and the last system act S_*. The user utterance U is converted to the feature vector where each element x_i is the count of occurrence of i^{th} n-gram in the utterance, where n ranges from 1 to 3. A set of context features s_i are extracted from the last system act, which is physically similar to user-act in the form of DA-type followed by a set of slot-value pairs, e.g. (DA-type), (DA-type, slot), (DA-type, slot, value) and (slot, value) [194]. Finally, the context features z_i are concatenated with n-gram features x_i to obtain the final utterance vector.

Thus the probability of a DA D of type $DA\text{-}type_j$ with a set of slot-value pairs $sv \in S$ on a user response u can be approximated by:

[5] SLU-Decoder : https://gitlab.cs.uni-duesseldorf.de/general/dsml/pydial3-public/-/tree/master/semi/CNetTrain

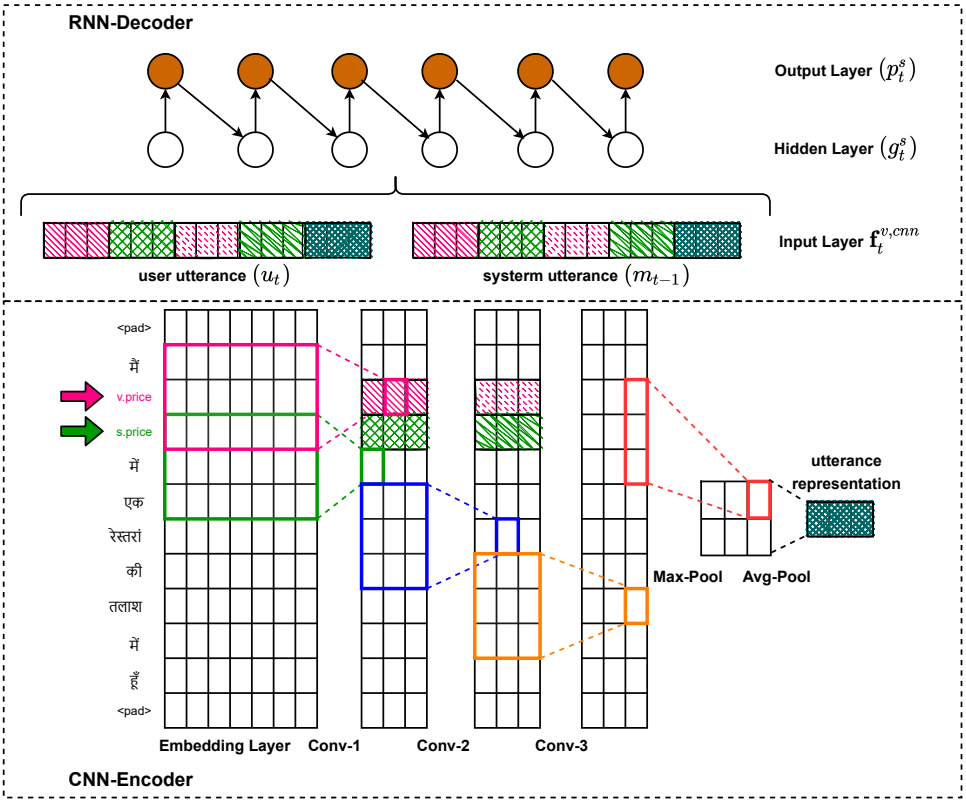


Figure 3.8 Architecture of RNN belief tracker with delexicalised CNN feature extractor.

$$P(D|u) = P\big(DA\text{-}type_j|u\big) \prod_{sv \in S} P(sv|u) \prod_{sv \notin S} \big(1 - P(sv|u)\big) \tag{3.1}$$

where, $P\big(DA\text{-}type_j|u\big)$ is the probability j^{th} DA-type on user input u, $P(sv|u)$ denotes probability of a slot-value pair sv. Both are generated by the classifiers trained earlier. For the resulting dialogue act to make sense, they must follow certain validity constraints:

- For DA-types *request*, it must contain at least one unbounded slot, e.g. (`address`), (`phone`).
- For DA-types *inform*, it must contain a bounded slot, e.g. (`price range`=सस्ता).

Belief Tracking

Based on the decoded output of SLU, a simple rule-based approach can be utilised to track the dialogue state. Two baseline trackers are used in the current work: (1) One-best (Basic) Baseline, (2) Focus Baseline [58]. For each component of the dialogue, i.e. turn-goal, turn-requests, the *basic tracker* keeps a single hypothesis whose value is highest up to the current turn. This simple non-statistical tracker has limitations of not accumulating evidence and goal-constraints from the past turns. The *focus baseline* is made to deal with these challenges by integrating the capability of evidence accumulation and handling the change of goal-constraints. As both are rule-based trackers, they are not scalable to the larger and dynamic ontologies.

3.4.2 RNN belief tracker with delexicalised CNN feature extractor

It is one of the first neural-based belief trackers [109] based on Henderson et al. [97]. At each turn, the belief tracker takes delexicalised form of current user utterance and last system response, extracts the CNN derived features and concatenate them to obtain the actual input feature vector as in:

$$\mathbf{f}_t^{v,cnn} = \text{CNN}_{s,v}^{(u)}(u_t) \oplus \text{CNN}_{s,v}^{(m)}(m_{t-1}) \tag{3.2}$$

where, u_t, m_{t-1} are the current user and previous system utterances which are passed through the slot-value specialised CNN operator $\text{CNN}_{s,v}^{(\cdot)}(\cdot)$, where each token is represented by an embedding of size N determined from a one-hot input vector. The CNN operator not only transforms the {user,system}-utterance into the encoded representation but also helps in extracting n-gram-like embeddings for delexicalised slots and values based on their position in the utterance through concatenating the Conv-1 and Conv-2 layer's output, as shown in Figure 3.8.

For each user input, the belief tracker's job is to maintain a multinomial distribution over values $v \in V_s$ for each information slots s as well as a binary distribution for all requestable slot-variables[6]. Thus, in order to track the occurrences of slots and its possible values, each slot in the ontology[7] requires its own specialised tracker. In [109], the slot-specific tracker is implemented by the Jordan-type RNN over the CNN's encoded features [213]. The probability $\mathbf{f}_t^v$ of each value v for a slot s is estimated from the corresponding RNN weights which takes inputted the feature vector $\mathbf{f}_t^{v,cnn}$ concatenated with the last turn context in each iteration (turn t):

$$\mathbf{f}_t^v = \mathbf{f}_t^{v,cnn} \oplus p_{t-1}^v \oplus p_{t-1}^\phi \tag{3.3}$$

where, p_{t-1}^v is the probability of value v expressed for slot s in the last turn $t-1$ while p_t^ϕ denotes the probability that the slot s is not mentioned by the user upto turn t. Collectively, the probability of a value v at turn t is estimated through softmax on the updated pre-softmax activation g_t^v as below:

$$p_t^v = \frac{exp(g_t^v)}{exp(g_{\phi,s}) + \sum_{v' \in V_s} exp(g_t^{v'})} \tag{3.4}$$

$$g_t^v = \mathbf{w}_s \cdot \sigma(\mathbf{W}_s \mathbf{f}_t^v + \mathbf{b}_s) + b_s^{'} \tag{3.5}$$

where, p_{t-1}^ϕ could be estimated by substituting $g_{\phi,s}$ for g_t^v in Equation 3.4. And vector $\mathbf{w}_s$, matrix $\mathbf{W}_s$, bias terms $\mathbf{b}_s$ and $b_s^{'}$, and scalar $g_{\phi,s}$ are parameters.

[6] diff between information and requestable slots.
[7] mentioned above.

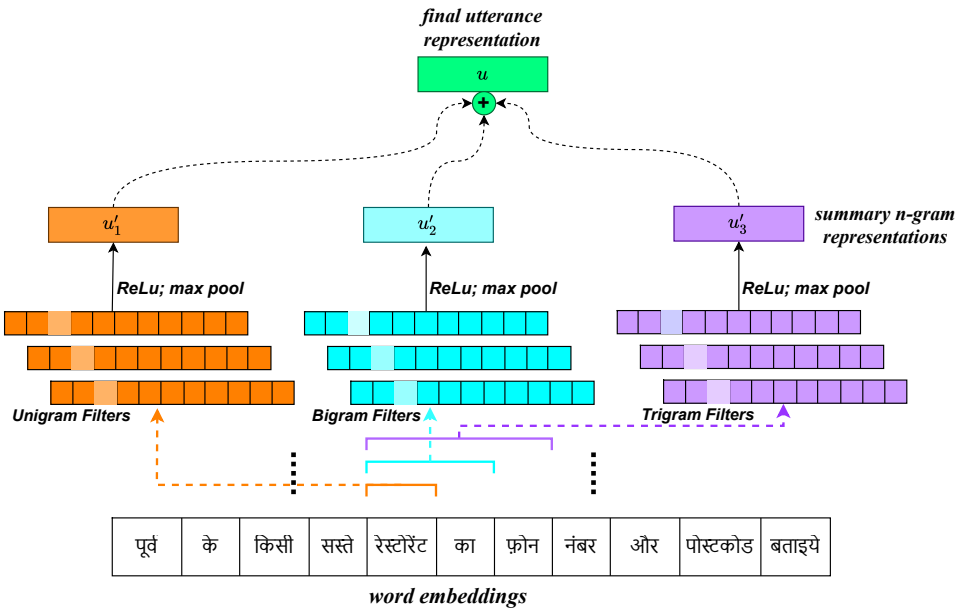


Figure 3.9 CNN-encoder to transform a user utterance by three convolutional filters which extracts uni-gram, bigram and trigram features.

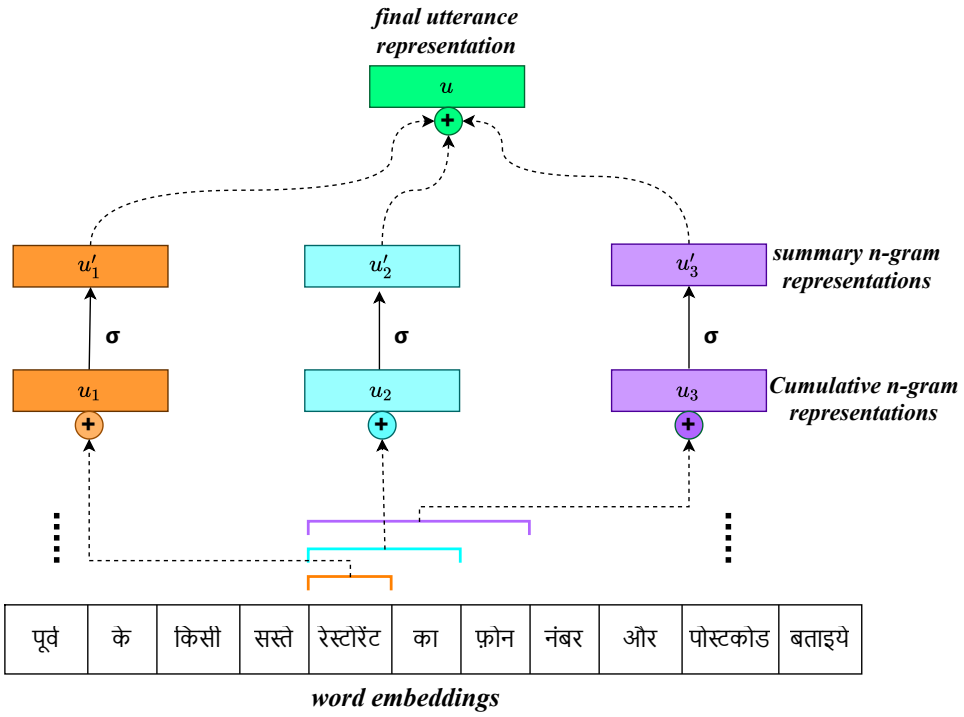


Figure 3.10 DNN-encoder to transform a user utterance into distributed representation through deep neural layers.

The delexicalised belief tracker performance is highly dependent on a manually developed *semantic dictionary* to identify an ontology term with all its lexical and morphological variations. Such coupled models are not scalable to larger and dynamic dialogue domains.

3.4.3 NBT-{CNN/DNN}

Removing the limitations of delexicalisation and dependency on domain-specific paraphrasing, NBT-{CNN/DNN} was proposed in [100] with additional capability of leveraging semantic information from pre-trained word vectors. The NBT model first encodes all the inputs to their corresponding intermediate representation. The current user response U is encoded to a distributed representation $\mathbf{u}_t$ by a CNN-encoder. The last system actions S_*, candidate slot-value pair SV are converted to their word embeddings $\mathbf{s}_t$ and $\mathbf{c}_t$. In Figure 3.9, CNN-encoder is shown, which performs convolution on a word-sequence with three parallel n-gram *filters*, i.e. unigram, bigram, trigram. The *context details* $\mathbf{m}_t$ (previous turn) and *semantic information* $\mathbf{d}_t$ (current turn) are obtained by the interaction of system-act representation $\mathbf{s}_t$ and candidate slot-value pair $\mathbf{c}_t$ respectively with the current user utterance $\mathbf{u}_t$ which together used to make *binary decision* $\mathbf{y}_t$ about the current slot-value pair.

Finally, getting the possible candidate slot-value pairs uttered by the user are joined to previous belief state $\mathbf{b}_{t-1}$ to obtain the updated belief state $\mathbf{b}_t$:

$$\mathbf{b}_t^s = \psi(\mathbf{y}_t^s, \mathbf{b}_{t-1}^s) \tag{3.6}$$

Here, $\mathbf{y}_t^s$ is a vector, consists of probabilities of all the values $v \in V_S$ for slot s. There are three ways to implement belief-state-update function ψ in NBT [100, 24], where the value-specific *one-step markovian update* method deliver higher accuracy in our case. In this way, the NBT model checks for each candidate slot-value pairs (exists in the ontology), and find out which have just been expressed in the user response. We have also shown the results of belief tracking using a *DNN-encoder*, which transforms the user utterance through two hidden layers: 1) *cumulative* n-gram *representation layer*, 2) summary n-gram representation layer into a distributed representation (see Figure 3.10).

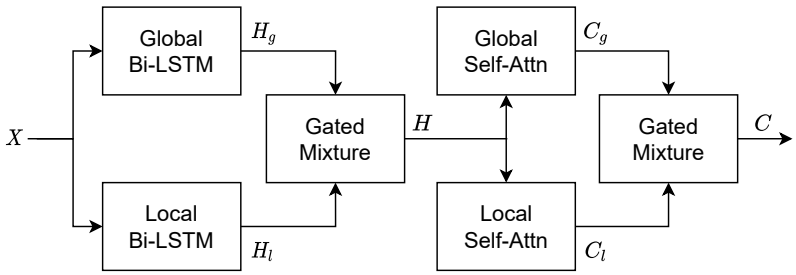


Figure 3.11 GLAD-Encoder. H, C are hidden and contextual representations for input X, i.e. U, S_i or SV.

3.4.4 GLAD-DST

The GLAD model also performs the DST task based on learning multiple binary classifiers for each slot-value pair. The poor detection of rare slot-value pairs in a turn, causes erroneous state tracking. This DST model resolves this issue by encoding the inputs user-utterance and previous system actions as well as slot-value under

consideration with three separate GLAD encoders before inputting them to the *decision-making* and *belief-updation* layers.

As shown in Figure 3.11, the GLAD encoder consists of two Bidirectional-LSTM (Bi-LSTM) [17] to capture temporal features in the input through a global Bi-LSTM for sharing parameters between each slot and a local Bi-LSTM to capture slot-specific features. The temporal features in the form of encoding are then summarised through self-attention to extract contextual features for the variable-length input sequence necessary for NLP tasks [214, 215]. The attention is also applied both globally as well as locally and then combined to produce summarised context. For each input type, the encoder constructs both the hidden representation and context summary:

$$\begin{aligned} H_{\text{utt}}, C_{\text{utt}} &= \text{encode}(U) \\ H_{\text{sact}_i}, C_{\text{sact}_i} &= \text{encode}(\text{S}_i) \\ H_{\text{sval}}, C_{\text{sval}} &= \text{encode}(\text{SV}) \end{aligned} \tag{3.7}$$

where, H_* and C_* denote the hidden encoding and self-attention context of corresponding user utterance U, i^{th} system action S_i and candidate slot-value pair SV to be evaluated (e.g. `food`=गुजराती)).

These encodings are processed by the decision-making and belief-updation layer to achieve an updated belief state for the current turn over the dialogue. In GLAD-DST, the decision-making layer computes two scores; *action score* (context-modelling) and *utterance score* (semantic modelling), that combinedly predicts the probability distribution on the candidate slot-value pair. The GLAD needs to learn different parameters to incorporate slot-specific information during the computation of temporal and context vectors. Thus, the lack of a unified encoder is the major limitation of GLAD-DST.

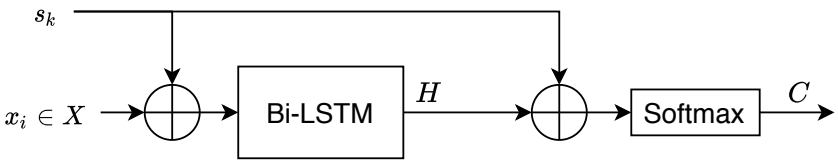


Figure 3.12 GCE-Encoder. H, C are hidden and contextual representations for input X, i.e. (U, s_k), (S_i, s_k) or (SV, s_k).

3.4.5 GCE-DST

Globally-Conditioned Encoder (GCE) based DST model is proposed as an improvement over GLAD architecture [103]. Instead of training different encoder for each slot, GCE employs an encoder with global conditioning on the embedding vector of slot-type, i.e. *food*, *area* or *price range*. Thus the model becomes computationally

less complex than the GLAD-DST. Except for the encoder, the rest of GCE-DST architecture is the same as GLAD-DST.

In place of slot-dependent global as well as local recurrent and self-attention layers, GCE uses only embedding vector of candidate slot s_*, as a conditioning vector to obtain temporal and contextual encodings as shown in Figure 3.12. For k^{th} slot, temporal representation H^k of input sequence X, i.e. user utterance or previous system actions, is computed through Bi-LSTM as below:

$$\begin{aligned} H^k &= \text{Bi-LSTM}(X \oplus s_k) \\ C^k &= \text{Self-Attn}(H^k) \end{aligned} \tag{3.8}$$

where, $\oplus$ is concatenation operator. s_k is the word-embedding of k^{th} slot to make decision about. Self-Attn() learns attention parameters to extract contextual summary on the temporal features obtained earlier.

The GCE-encoder encodes the user utterance, previous system action and the candidate slot-value pair as follows:

$$\begin{aligned} H^k_{\text{utt}}, C^k_{\text{utt}} &= \text{encode}(U, s_k) \\ H^k_{\text{sact}_i}, C^k_{\text{sact}_i} &= \text{encode}(\text{S}_i, s_k) \\ H^k_{\text{sval}}, C^k_{\text{sval}} &= \text{encode}(\text{SV}, s_k) \end{aligned} \tag{3.9}$$

The encoded features are given to decision-making and belief-updation layer to predict belief state same as in GLAD-DST model.

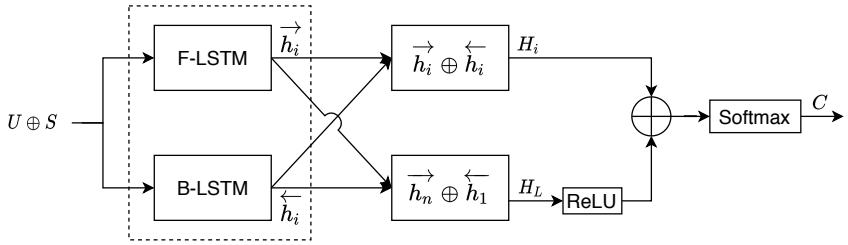


Figure 3.13 GSAT-Encoder. H, C are hidden and contextual representations for input $U \oplus S$.

3.4.6 GSAT-DST

Although GCE simplified the GLAD architecture and limits the computational complexity up to a level, it still has a substantial time complexity for real-world applications. This is due to the fact that both GLAD and GCE use separate recurrent encoding modules processing user utterance, system actions and candidate slot-value pair individually to generate their output representation.

Global encoder and Slot-ATtentive decoder (GSAT) is proposed to overcome this computational complexity and improve the prediction latency time [105] with maintaining state-of-the-art accuracy. This model carries an encoder module and a set of slot-specific classifier (decoder) modules.

Unlike GLAD and GCE, GSAT-encoder takes the user-utterance U and a set of system-actions S together and generates slot-specific input representation H_s context vector C_s in one go (see Figure 3.13) [105].

$$\begin{aligned} H_s &= \text{Bi-LSTM}(U \oplus S) \\ C_s &= \text{Self-Attn}(H_s) \end{aligned} \tag{3.10}$$

In place of modelling semantic and contextual details separately, GSAT-DST trains a decoder (classifier) Z_s to obtain the value distribution V_s of slot s as:

$$Z_s = W_s V_s \tag{3.11}$$

where, $V_s = \{v_1, v_2, ...\}$ is distribution of values for slot s and W_s are trainable parameters. $softmax(C_s \cdot Z_s)$ is used to obtain the distribution of informable slots S_{inf}, whereas $sigmoid(C_s \cdot Z_s)$ derives the requestable slots S_{req}. Taking input in one go through the encoder as well as decoder remarkably faster in both training as well as testing and hence improves DST's overall performance.

3.4.7 Simple-BERT DST

Most of the Neural Belief Trackers have much complex architecture, leading to difficulty in implementing, maintaining, and debugging the code. Most of them are none operative in the situation when the ontology is dynamically changed. The Simple-BERT DST model [107] is very effective in handling the above issues. Also, the number of parameters in this model does not grow with the size of ontology (i.e. increasing the values associated with the slots in the ontology).

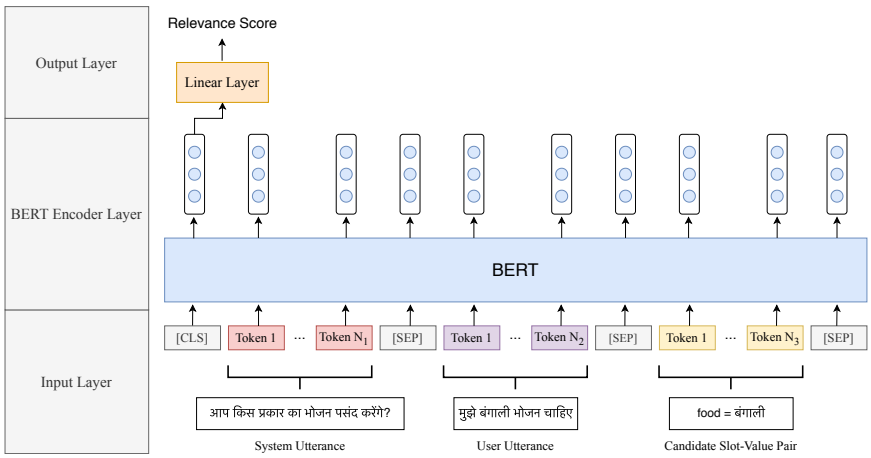


Figure 3.14 Simple-BERT DST Architecture.

The model architecture consists of two parts:

1. *Encoding Layer*: The input sequence (I_t) to this layer is a sequence of system-user utterance followed by candidate slot-value pair, as shown in Figure 3.14. The input sequence is sent to the pre-trained BERT model. The input I_t is represented as:

$$I_t = \texttt{[CLS]}\ S_t\ \texttt{[SEP]}\ U_t\ \texttt{[SEP]}\ C_t\ \texttt{[SEP]} \tag{3.12}$$

   Where S_t is system utterance, U_t is user utterance, and C_t is slot-value pair. The input token sequence sent to the BERT involves using `[CLS]` and `[SEP]` tokens as BERT-specific classification token and separator token, therefore X_t looks as follows:

   X_t = `[CLS]` <Sys-utt> `[SEP]` <User-utt> `[SEP]` <s-v pair> `[SEP]`

   For example:

   X_t = `[CLS]` आप किस प्रकार का भोजन पसंद करेंगे `[SEP]` मुझे बंगाली भोजन चाहिए `[SEP]`
   food=बंगाली `[SEP]`

   BERT is a language representation model that uses a multilayer of Transformers [214]. The pre-trained BERT model [102] trained over large unlabelled corpora is used for encoding. It produces an output representation corresponding to each input token. The first `[CLS]` token is a special classification token, and the output vector obtained by the BERT corresponding to it is used as an aggregated representation of the sentence.

2. *Decision Making Layer*: The encoded output vector obtained corresponding to `[CLS]` token (i.e. h_0) is passed through the neural layer with sigmoid activation function:

$$y = \sigma(Wh_0 + b) \tag{3.13}$$

   where W and b are model parameters corresponding to each slot-type. The candidate slot-value pair is relevant if the output of the network (y) is at least 0.5

The belief state is updated based on this new slot-value pair prediction. Example: Let us suppose that in the current turn, the slot-value pair `food=`'बंगाली' is predicted. If the dialogue state does not have a value associated with *food* in the belief state, then `food=`'बंगाली' will be added to the belief state, but in case the slot-value pair `food=`'पंजाबी' already exists, then, in that case, the belief state is updated with `food=`'बंगाली'.

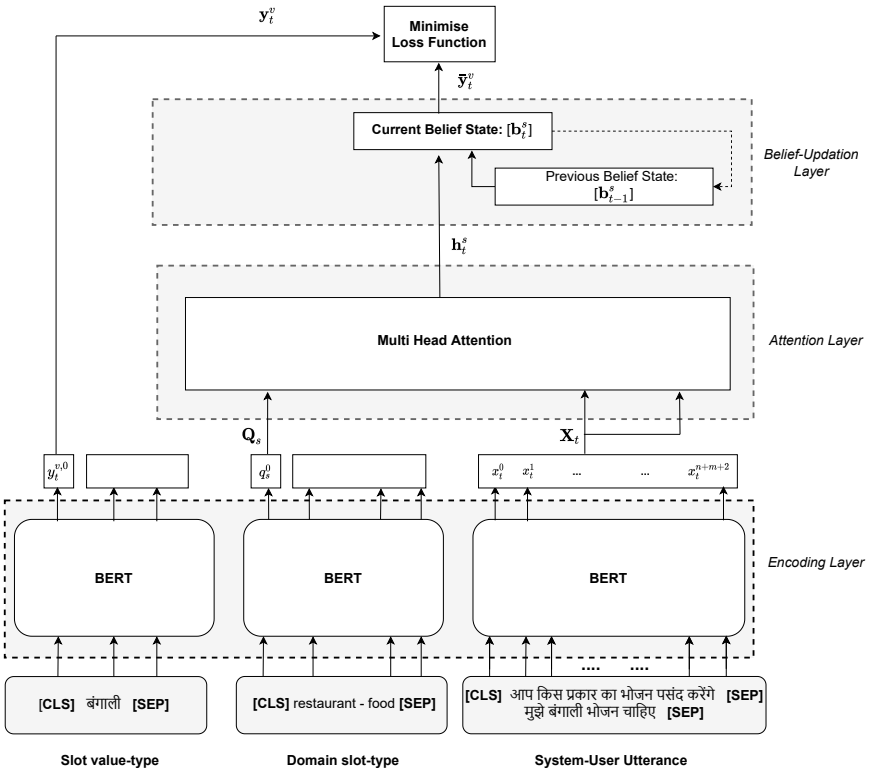


Figure 3.15 SUMBT Architecture.

3.4.8 SUMBT

The neural belief trackers discussed so far have modelled the trackers that are domain and slot dependent. They hold a major drawback to not adding extra slot values to ontologies, and it leads to creating a rigid domain ontology setup [216]. This problem is tackled by Slot-Utterance Matching for universal and scalable Belief Tracking (SUMBT) [108]. It uses the attention mechanism over the user utterance for learning the appearance of domain slot-type and slot-values. For example: In our case, consider a domain slot-type ('restaurant-area'), the SUMBT finds the appearance of slot-value ('centre') type in a pair of system and user utterances. The attention mechanism uses a contextual semantic vector formed through BERT [102].

The architecture of SUMBT contains three components (shown in Figure 3.15):

1. *Encoding Layer:* It is the first layer of the model where the pre-trained BERT model [102] is applied to obtain contextual semantic vectors. Using the BERT encoder, the following encoded vectors are obtained for turn t: system-user utterance vector (X_t), domain slot-type vector (Q_s) and target slot-value vector (i.e. y_t^v).

$$X_t = \text{BERT}([S_t \oplus U_t]) \tag{3.14}$$

2. *Attention Mechanism Layer:* The multi-head attention [214] is performed that uses domain slot-type vector (Q_s) as a query matrix Q. It focusses on user-system utterance vector (X_t) represented as a key

matrix K and value matrix V. These attention vectors is represented by (h_t^s) using:

$$h_t^s = \text{MultiHeadAttention}(\text{Q, K, V}) \tag{3.15}$$

3. *Belief-State Updation Layer:* The previous belief state (b_{t-1}^s) and the current turn (h_t^s) of the dialogue are needed to model the current belief state (b_t^s) of the turn (derived in Equation 3.16). RNN, LSTM, GRU or Transformers can be used as a modelling function (ψ). The belief state obtained is then passed through normalisation layer to output the predicted slot-value vector ($\bar{y}_t^s$):

$$b_t^s = \psi(b_{t-1}^s, h_t^s) \tag{3.16}$$

The model is trained to minimise the distance between the predicted slot-value ($\bar{y}_t^s$) vector and the target slot-value (y_t^v)vector. The distribution corresponding to slot-value (v_t) is shown as:

$$p(v_t|X_t, s) = \frac{e^{-d(\bar{y}_t^s, y_t^v)}}{\sum\limits_{v \in C_s} e^{-d(\bar{y}_t^s, y_t^v)}} \tag{3.17}$$

The distance used are *Euclidean distance* and *Cosine distance*. At last, the model is trained using log-likelihood based loss estimation (Equation 3.18), where s and t represent domain slot-type and dialogue turn, respectively:

$$\mathcal{L} = -\sum_{s \in D} \sum_{t=1}^{T} \log p(v_t|X_t, s) \tag{3.18}$$

3.5 Experiments

3.5.1 Word-Embeddings

We use the AI4Bharat-IndicNLP[8] Corpus to train various word-embeddings models for the experiment [217]. It is a set of large-scale, general-domain monolingual corpora for 10 Indian languages. Hindi monolingual corpora, consists of 62,961,411 sentences and 5,322,594 types of unique words, is collected mainly on news domain and Wikipedia that covers contemporary use wide range of topics on the Hindi language.

In this work, we use fixed, non-contextual word-embeddings, i.e. Word2Vec, GloVe, FastText. *Word2Vec* has established the state-of-the-art of word-embeddings in NLP tasks and showed the way to train word vectors through neural-networks [218] based on local context window methods. The GloVe makes the efficient use of statistics on global word-to-word co-occurrence matrix [219]. On the other hand, *FastText* is capable of

[8] https://github.com/AI4Bharat/indicnlp_corpus

integrating subword information in the form of *character* n-gram embeddings [220] which is beneficial for morphologically rich languages. Standard embedding size 300 is applied to all word-embedding models.

We train both *CBOW* and *SG* word-embedding models over Word2Vec as well as FastText. Based on the suggestions in [217, 221], the training hyper-parameters are set for ten epochs with a window size of 5, minimum token count of 5 and negative sampling of 10 for both Word2Vec and FastText models. For the GloVe model, the same parameter setting for training iterations, window-size, minimum token count are used. To investigate the behaviour of different DST models without any pre-trained embeddings, we utilise XAVIER (random) word-vector for the evaluation.

3.5.2 Metrics

Accuracy of the SLU models are estimated on precision, recall and F-1 score where *precision* refers to the percentage of the results which are relevant, *recall* refers to the percentage of total relevant results correctly decoded by the semantic decoder. If X is the set of reference output and Y is denotes the set of predicted output, the F-1 score is calculated by:

$$\text{F-1 score} = \frac{2|X \cap Y|}{|X| + |Y|} \tag{3.19}$$

where, $X = (DA\text{-}type_{ref} \cup sv_{ref})$

$Y = (DA\text{-}type_{out} \cup sv_{out})$

The DST models are evaluated on *joint-goal* and *turn-request* accuracy. During the evaluation of each turn, the joint-goal is obtained through accumulating the turn-goals. In the belief tracking process, current turn-goal specification takes precedence over the previously specified value for a slot. For example, suppose the user mentions `food=`गुजराती in the current turn. If the slot *food* has not been specified before, then the `food=`गुजराती turn-goal would be added to the joint-goal. Otherwise, any previous specification (such as `food=`मराठी) is replaced by it.

Table 3.4 Implementation details for various DST models. The Learning-Rate is used with *Adam* optimiser.

Models	Learning Rate	Dropout	Batch Size	Epochs
NBT-DNN	1e-3	0.5	256	400
NBT-CNN	1e-3	0.5	256	400
GLAD	1e-4	0.2	50	400
GCE	1e-4	0.2	50	400
GSAT	1e-3	0.2	32	150
Simple-BERT	2e-5	0.1	16	25
SUMBT	5e-5	0.1	4	300

3.5.3 Implementation Details

Only NBT-{CNN/DNN} is implemented using the *TensorFlow* [222], and the rest are based on the *PyTorch* library [223]. Word-embedding size for each DST model belong to category-1 is set 300. All the DST are compared on a set of pre-trained word embeddings, i.e. GSAT-{CBOW,SG}, FastText-{CBOW,SG} and GloVe. All DST-models hyper-parameters are tuned on a separate validation set. In order to have a fair comparison, all the DST models are experimented with 10 different random initialisation, and the accuracies shown in Section 3.6 are the mean of them.

Table 3.4 shows the implementation details of various DST models. Both NBT-CNN and NBT-DNN are trained by Adam optimiser (lr=0.001) with 0.5 dropout, and batch-size 256 for 400 epochs, and loss is estimated through softmax cross-entropy with logits similar to [24]. On the other hand, GLAD and GCE are optimised by Adam (lr=0.0001) with 0.2 dropout to learn the hyper-parameters for 400 epochs, and 50 batch-size and binary cross-entropy is used to calculate the loss for both [103, 104]. Similarly, GSAT also uses Adam optimiser (lr=0.001) with 0.2 dropout between the layers. It learns through the batch-size of 32 for 150 epochs.

In category-2, a pre-trained multilingual BERT model[9], having 12 layers of 784 dimension and 12 attention heads, is used in both Simple-BERT DST and SUMBT models. Simple-BERT DST is trained using BertAdam optimiser (lr=2e-5) with batch-size 16 for 25 epochs, and loss is estimated through cross-entropy. On the other hand, SUMBT employs the configuration of 4 multi-head attention of 784 hidden size. In the belief tracker, a single-layered LSTM, GRU and Transformer with the hidden size of 300 are employed. The maximum input sequence length of 64 is chosen. This model is also trained using BertAdam optimiser (lr= 5e-5) with batch-size 4 for 300 epochs, and loss is estimated using the Euclidean distance metric.

Table 3.5 Precision, Recall and F-1 score of SLU-models.

SLU-Models	Precision	Recall	F-1 score
SGD	83.15	33.48	47.74
SVM	91.52	34.06	49.57
Delexicalised-CNN+RNN	**98.15**	**57.43**	**72.46**

3.6 Results & Discussion

First, we show the performance of baseline SLU models on our corpus. Table 3.5, presents the results obtained through SVM/SGD decoders and delexicalised-CNN based SLU models. It is evident that the neural-based semantic decoders are more capable of detecting the DA-type and slot-values from a user utterance than the simple tuple classifiers.

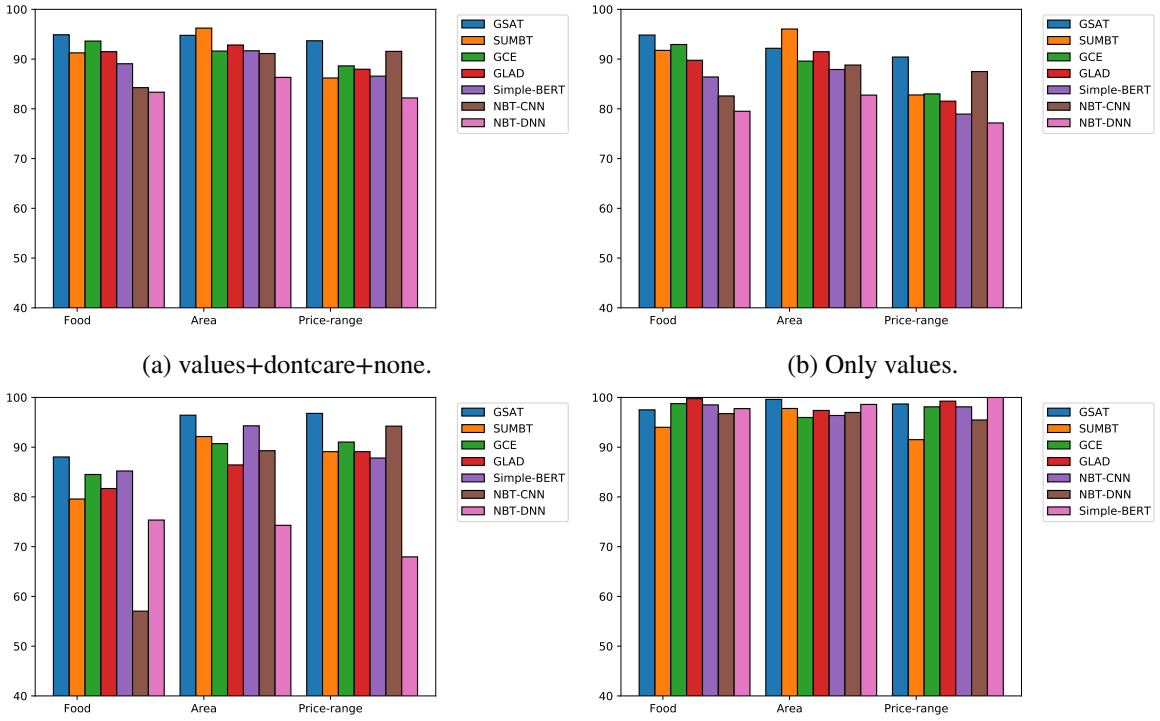
[9]Bert-base-multilingual-cased : https://huggingface.co/bert-base-multilingual-cased

Table 3.6 Comparison of all DST models on Joint-Goal accuracy.

Category	DST-Models	Joint-Goal
Baselines	SGD-Focus	66.61
	SGD-Basic	62.63
	SVM-Focus	67.96
	SVM-Basic	62.37
	Delexicalised-CNN+RNN	**72.40**
Category-1	NBT-DNN	61.50
	NBT-CNN	69.00
	GLAD	74.71
	GCE	74.52
	GSAT	**83.25**
Category-2	Simple-BERT	68.75
	SUMBT+Transformer	72.40
	SUMBT+LSTM	75.14
	SUMBT+GRU	**77.14**

Table 3.7 Comparison of Category-1 DST models on joint-goal accuracy. Where, W2V=*Word2Vec*, FT=*FastText*, CBOW=*Continuous Bag-of-Words*, SG=*Skip-Gram*

DST-Models	XAVIER	GloVe	W2V-CBOW	W2V-SG	FT-CBOW	FT-SG
NBT-DNN	42.70	53.90	52.90	54.90	57.40	61.50
NBT-CNN	50.50	59.10	64.10	67.30	66.67	69.00
GLAD	51.67	63.65	66.87	72.83	68.33	74.71
GCE	52.31	65.11	68.45	72.10	70.52	74.52
GSAT	56.78	72.16	69.48	75.62	72.40	**83.25**



(a) values+dontcare+none.

(b) Only values.

(c) Only dontcare.

(d) Only none.

Figure 3.16 Slot-accuracy comparison of the DST models on prediction of (a) values+dontcare+none, (b) values, (c) dontcare and (d) none.

Table 3.8 Comparison of Category-1 DST models on turn-request accuracy. Where, W2V=*Word2Vec*, FT=*FastText*, CBOW=*Continuous Bag-of-Words*, SG=*Skip-Gram*

DST-Models	XAVIER	GloVe	W2V-CBOW	W2V-SG	FT-CBOW	FT-SG
NBT-DNN	80.70	79.50	82.10	71.20	73.20	81.60
NBT-CNN	85.60	83.50	86.50	87.00	75.80	91.50
GLAD	95.44	95.01	95.56	95.74	95.56	95.38
GCE	95.02	94.95	94.77	95.08	96.35	94.77
GSAT	95.08	96.05	95.01	95.38	96.11	96.17

The *joint-goal* performance of the category-1 models is shown in Table 3.7 with a comparison on various word embeddings. We can observe that there is an evident influence of embeddings on the joint-goal accuracy. FastText based skip-gram embedding shows higher performance. The reason is that FastText-SG performs well for a morphologically-rich language as it takes into account the internal structure of words while learning word representation [220]. From the table, we also observe that the DST models trained on skip-gram based word embeddings comparatively achieve higher accuracy as it captures rare words better than CBOW or GloVe. Whereas, XAVIER embedding has obtained the lowest accuracy as it doesn't have the capability to show any proximity between two similar words.

Among all, GSAT has obtained the highest joint-goal accuracy consistently on all embeddings as it jointly encodes the user-input, system-actions and candidate slot-value pairs with better hidden representation and self-attention layers. Performance-wise, GLAD and GCE are close to each other. NBT with CNN-encoder gives better results over the DNN-encoder as its convolutional filters generates a better representation of user utterance.

The Category-2 models are more robust towards new slot values compared to Category-1. It is being verified by tracking the performance of food slots which were not present in the training and validation dataset, such as: 'देशी' (Deshi), 'लखनवी' (Lakhnawi), 'आलू भरे पराठे' (Aalu bhare parathe)in the testing dataset. On comparing the joint-goal accuracy of both models, the GSAT (Category-1) shows the accuracy of 0%, whereas Simple-BERT DST (category-2) models shows an accuracy of 62.5%. The GSAT (Category-1) can not predict any of the newer slot values as the model uses the static ontology, and hence the output corresponding to the newer slot value is worse. On the other hand, Simple-BERT DST performed quite well in such predictions. However, the dynamic-ontology-based models make some mistakes in capturing the slots whose values are "dontcare".

On comparing the joint goal accuracy of the models in category-2 (see Table 3.6): SUMBT outperforms the Simple-BERT DST. The SUMBT has an attention mechanism working over the encoded system-user utterance that focusses on the domain slot-type and slot-values. It led to improved performance of the SUMBT model.

Table 3.8 refers to the turn-request performance of the category-1 models. It is observed that GLAD, GCE and GSAT are close to each other in predicting the request slot with an approximate accuracy of $\approx 95\%(\pm 1)$. The reason is that predicting a requestable slot is much easier than predicting an informable slot due to infrequent

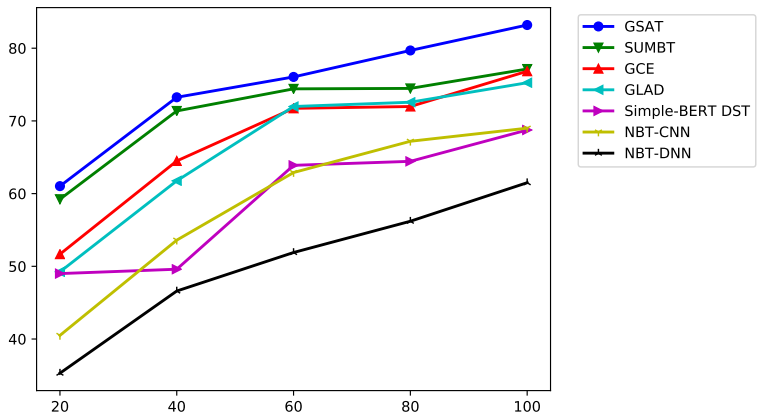


Figure 3.17 Comparing the joint-goal accuracies of DST models on various scale of training data sampled at {20%, 40%, 60%, 80% and 100%}.

and low-variation of request slot-value in a turn. In contrast, the NBT-{CNN/DNN} obtain low accuracy in turn-request prediction.

In order to facilitate the comparison of the results for DST models trained on the different amount of training data, the joint-goal accuracy curves are plotted in Figure 3.17. As the training-data size increases, the model accuracies increases monotonically indicating the importance of size of corpus on the accuracy. GLAD and GCE models are here also close to each other. GSAT is found to be higher than the others on each scale. For a smaller training dataset, the plot of SUMBT performance is closer to GSAT.

Now, we compare the performance of DST models in predicting the different slot-values (i.e. dontcare, none, values) as shown in Figure 3.16. GSAT achieves highest slot-accuracy for any slot-value combination. Based on the decreasing order of slot accuracies in Figure 3.16a-3.16b, the DST models can be ordered in the sequence: GSAT>SUMBT>GCE>GLAD>Simple-BERT>NBT-CNN>NBT-DNN. For the *dontcare* value, the slot-accuracies of the models corresponding to food slot are comparatively lower than the *area*, and *price range* as the distribution of values in *food* is higher compare to *dontcare*. The slot-accuracy in predicting the *none* value is shown in Figure 3.16d where the models are showing similar performance.

Table 3.9 Average time required in one epoch for each DST model (in seconds).

DST-Models	**Avg Time per epoch (sec.)**
NBT-DNN	420
NBT-CNN	35.5
GLAD	150
GCE	100
GSAT	**4.5**
Simple-BERT	5827
SUMBT+GRU	175.5
SUMBT+LSTM	178
SUMBT+Transformer	202

All the models are executed under the same environment and hardware (single Nvidia Quadro K6000 12GB GPU). Based on the implementation and approach, the pre-processing and post-processing of each model can vary. Hence, we compare the models only on the time taken to execute in one epoch after it is loaded and ready to be executed. The Average time taken in one epoch for all the models is shown in Table 3.9. As NBT-{CNN,DNN} train separate tracker for each slot, their time-complexity needed to be estimated for each slot individually. For NBT-DNN, per-epoch average times are 90, 134, 96 and 100 seconds respectively for request, food, price range and area slots. Similarly for NBT-CNN, these times are 7, 13, 7.5 and 8 seconds. Hence, NBT-DNN and NBT-CNN take approximately 35.5 and 420 seconds per epoch. Moreover, GLAD, GCE and GSAT take 150, 100 and 4.5 seconds respectively per epoch under the same hardware configuration. GSAT takes the least amount of time for training one epoch due to its fast encoding process. On the other hand, Simple-BERT takes a large amount of time to train a single epoch as it generates and processes negative examples to the size of ontology in each turn.

Table 3.10 Comparison of HDRS and WOZ 2.0 during the training with and without a language specific pre-trained embedding on joint-goal (%) accuracy.

HDRS (Hindi)		**WOZ 2.0 (English)**	
Embedding Layer	**FastText (IndicNLP Corpus)**	**Embedding Layer**	**FastText (Common Crawl)**
56.80	83.25	88.70	87.30

Handling morphological properties is one of the major issues in building natural language processing applications in the Hindi language [44]. To prove the argument, we performed an experiment where the GSAT-DST model is trained on both in Hindi & English corpora under two cases where: 1) Both are trained on language-specific pre-trained embeddings; 2) Both are trained without any pre-trained embedding with an inbuilt embedding layer in the GSAT model.

From Table 3.10, we can observe a significant difference in joint-goal accuracy of HDRS (Hindi) in comparison to WOZ 2.0 (English) when trained on with/without a language-specific pre-trained embedding. In WOZ 2.0, the joint-goal accuracy does not change much when using the pre-trained embeddings in place of non-pretrained embeddings. But, for HDRS (Hindi), there is a significant difference in the joint-goal accuracy when using pre-trained embeddings (FastText-IndicNLP). Because the FastText captures sub-word information that is critical for representing the information in a morphologically-rich language. Hence the performance for morphologically rich language, e.g. Hindi, will show a significant rise when using a pre-trained FastText embedding in place of non-pretrained embedding.

3.7 Summary

This chapter has described the proposed Hindi Dialogue Restaurant Search (HDRS) corpus and compared various state-of-the-art SLU/DST models on it. First, a brief discussion on the collection of HDRS corpus through WOZ and its various features with corresponding challenges is given, then compared various language understanding and dialogue state tracking models on it. Among 1.4k dialogues in the corpus, there are 40% dialogues where the user changes his/her goal. It signifies how natural and challenging the data is for the dialogue state tracking task.

The chapter has pointed out the significance of a DST architecture that jointly performs the task of language understanding and dialogue state tracking. To prove this, both SLU-detached and SLU-joint DST models are investigated on the proposed dataset. Neural models for the DST, i.e. RNN-CNN, NBT-{CNN/DNN}, GLAD, GCE, GSAT, Simple-BERT and SUMBT, have also been explored to show their performance. Here RNN-CNN, NBT-{CNN/DNN}, GLAD, GCE, GSAT are the Category-1 DST models which utilise explicit pre-trained embeddings such as GloVe, Word2Vec-{CBOW, SG}, FastText-{CBOW, SG}, while the Category-2 DST-models, i.e. Simple-BERT DST, SUMBT, use the pre-trained multilingual-BERT encoder. Category-2 can handle dynamic ontology; hence suitable for the dialogues where domain-ontologies get updated frequently. In addition, the performance of DST models on HDRS (Hindi) and WOZ 2.0 (English) is compared, and a significant difference in the joint-goal accuracy is observed when non-pretrained embeddings are used in both cases. It shows the importance of pre-trained embeddings in NLP tasks for morphologically rich languages, e.g. Hindi.

The next chapter will present the approaches of dialogue policy, a central component of the SDS pipeline, with a specific focus on modelling dialogue in terms of the dialogue state, the system's action and the reward under the reinforcement learning paradigm.

Chapter 4

Modelling Dialogue Management through Reinforcement Learning

4.1 Introduction

As defined in Section 2.4, it is the job of the dialogue manager to control the flow of the dialogue. A simple approach to realise this is defining a set of rules that the system would follow during the dialogue. Such systems are generally *system-directed*, where the dialogue manager attempts to control the discourse by asking the user question, which the user then answers [224]. Initially, some dialogue managers are based on a similar structure [15], uses VoiceXML[1] [225, 226] to implement the rule-based functional specification.

Researchers found an alternative approach to build a rule-based system known as frame-filling or form-filling [119]. Distinctively, it decouples rules for handling user input, e.g. *slots*, from those maintaining the dialogue flow, e.g. *fillers*. During the discourse, a collection of slots, called a *frame* or *form*, will be filled with the values provided by the user. Ultimately, the complexity of these systems is expressed in terms of slots and the number of values they support. It also supports the *user initiative* way of interaction [224].

The form-filling approach of rule-based systems is further extended to the agenda-based dialogue management framework [227]. It is more flexible and supports more complex dialogues scenarios. One major limitation of all rule-based systems is that they require a separate *error handler* to detect speech-understanding errors and provide a strategy to deal with them [228]. They also do not support the automatic optimisation of the dialogue manager.

Recently, it has been shown that statistical methods [121–124] can resolve the limitations of the rule-based approaches. Probabilistic methods in dialogue management can model uncertainty in the system's belief state and map it into a distribution over sets of system action. It helps the system to be more robust towards various

[1] VoiceXML: https://www.w3.org/TR/voicexml20/

noisy conditions. A dialogue policy can be implemented as a classification task that trains on dialogue corpus data. Such dialogue managers are highly portable and extendable across different domains.

However, supervised learning of dialogue management faces severe sparsity issues as the dialogue domains are usually exponential in the number of distinct instances they can generate. Even a very large dialogue corpus would represent only a tiny fraction of the total set of plausible dialogues. Hence, a significant amount of abstraction is required to limit the space of dialogue behaviour that can be learnt. However, leveraging such supervised behaviour does not guarantee that it would lead to a successful dialogue [125].

An alternative is to use Reinforcement Learning (RL), where the dialogue interaction is considered as a long-term planning task with optimising its action selection policy with respect to an objective measure [126]. Unlike the supervised learning models, where the dialogue manager's behaviour is restricted to the type corpus used, a dialogue manager using RL can explore all possible behaviour. Several statistical approaches have been utilised to learn the dialogue policy, i.e. Point-based methods [128, 2], Gaussian-based methods [124]. However, they are found to be effective for modelling relevant, reachable belief states, but they are unable to scale to sizeable state-action space.

As the number of possible dialogue states can be very large, complex and universal approximation functions, such as Neural Networks (NNs), i.e. Policy-gradient methods, Deep Q-Network (DQN), Deep Reinforcement Learning (DRL), have been used recently for dialogue policy modelling [130, 229, 230]. Although DQN has resolved the scalability issues of the dialogue policy learning with high convergence capability, they are typically slow gradient-based methods due to low sample efficiency. Advantage Actor-Critic (A2C) methods achieve better performance as they acquire the positive aspects of both value-based and policy-based methods. Due to based on the on-policy RL methods, such policy learning methods suffer from low sample efficiency [231].

In this chapter, we have explored and investigated the current state-of-the-art methods of policy optimisation for a task-oriented dialogue system. Inspired by [231], we present a new method that combines the strength of Experience-Replay (ER) in A2C policy learning for better dialogue modelling. As an *actor-critic*, it uses both value-based (critic) and policy-based (policy) functions for policy learning to handle high-variance data. We show that incorporating experience-replay not only makes the method sample-efficient (hence speed up policy learning) but also improve the overall success rate and episodic reward. To carry out the experiments, we adapted the *agenda-based user simulator* [122, 232] for the Indic language (i.e. Hindi) environment. The contributions lie in the following ways:

1. We incorporate the *Advantage Actor-Critic with Experience Replay* (A2CER) algorithm [231, 233] for dialogue policy learning which has recently been shown to be performing well on simple gaming environments.

2. We compare its performance with other state-of-the-art methods on a dialogue task with discounted (delayed) rewards in the Hindi language.

3. For better user experience in *on-line policy learning*, the models are demonstrated to utilise a demonstration data (HDRS [1]), achieving improved early-stage performance.

4. We also compare the performance of the methods through a human evaluation.

The rest of the chapter is organised as follows. We first discuss the use of Reinforcement Learning (RL) in dialogue management, exploring all possible behaviour in the dialogue (in Section 4.2). Then, Section 4.3 provides the overview of dialogue policy optimisation methods (e.g. value-based and policy-based methods) and the taxonomy of RL methods based on different dimensions and characteristics. Section 4.4 presents the proposed A2CER algorithm that describes the advantage of using ER on A2C policy learning method. Then, in Section 4.5, we describe experimental setups comprising domain-ontology, user simulator, reward estimation criteria and configuration of the models to be compared. The result and discussion of extensive evaluation is given in Section 4.6. Finally, the conclusions and future work directions are given in Section 4.7.

4.2 Reinforcement learning in dialogue management

RL is a subfield of machine learning whereby the agent (the machine) learns from interaction with the *environment*. In a situation, the agent *observes* the environment represented by a *state* and determines which *action* to take and receives a *reward*. The agent aims to take a sequence of actions that lead to the highest total (or expected) reward. In the last decade, many research works have shown the usefulness of the RL framework in dialogue applications [128, 234, 235, 2], especially under the framework of Partially Observable Markov Decision Processes (POMDPs). The definition and training methods of POMDPs are discussed in the following paragraphs.

POMDPs are considered to be the generalisation of Markov Decision Processes (MDPs) [236, 237]. It models an agent operating in a world in which it is assumed that the system dynamics are decided by an MDP, but the agent can not directly observe the underlying state. An MDP is defined as a mathematical framework formalising the interaction of an agent with a stochastic environment [238, 239]. Mathematically, it is represented by a tuple $\{S, A, T, R, \gamma\}$, where S denotes set of all states, the agent can be in, and A are possible actions, it can take, T: $P(s_{t+1}|s_t, a_t)$ represents the Markovian state transition function, R: $r(s_t, a_t)$ defines the immediate reward, and γ is a geometric discount factor for limiting the influence of future rewards during the cumulative reward estimation at the current state.

In the MDP setting, the goal is to find a policy π which selects an action at each state, $\pi : S \rightarrow A$, where the state in the environment is fully observable. Ideally, the policy's goal is to determine an action given on the

entire history so far, consisting of all previous states and actions. However, due to the high complexity of long sequences, it is often intractable in real situations. The state is assumed to satisfy the *Markov property* to handle this issue and bound to depend only on its previous state.

Considering as a stochastic process, the policy can either be conditional distribution $\pi(a|s)$ or deterministic, which is $\pi(s) = a$. In order to make a decision, the policy's objective is to maximise the expected discounted reward, R_t^π, which is the sum of discounted rewards from time t upto a potentially infinite horizon:

$$R_t^\pi = \sum_{i=0}^{\infty} \gamma^i r_{t+i+1} \tag{4.1}$$

where, the discounted factor $\gamma \in [0, 1]$ is used to reduce the importance of the reward received at the later steps. The MDP framework offers a model for dialogue management that not only makes a dialogue manager less dependent on the domain but also trainable from a given data. It can be used to build real-world spoken dialogue systems with enough approximations [127]. However, the MDP does not have the ability to deal with the corrupted Automatic Speech Recognition (ASR) [15], Spoken Language Understanding (SLU) [83, 1] and Dialogue State Tracking (DST) [1] outputs due to different levels of noisy conditions and the inherent ambiguity of the natural language. Its assumption on the full observability of the dialogue states prevents it from keeping track of alternative dialogue states during the dialogue. It leads to POMDPs, which models the dialogue state as a latent variable estimated on the noisy environment. Therefore, it offers a principled mathematical model for agents to perform in a non-deterministic way under partial observability, making it suitable for handling real-world sequential decision tasks.

A POMDP is defined as a 7-tuple $\{S, A, T, R, \Omega, O, \gamma\}$, where $\{S, A, T, R, \gamma\}$ is the underlying MDP, Ω is the set of *observations* (alternative states), and O: $P(o_{t+1}|s_t, a_t)$ is the observation probability. It can be represented as a Dynamic Bayesian Network, as given in Figure 4.1. All the tuple variables and connections between them in the dialogue scenario are discussed below:

- **State**: A *state* $s \in S$ consists of all the relevant information captured from the environment (user). In contrast to MDP, the states are hidden in POMDP and must be inferred from the observation $o \in O$. Due to its dependency on the number of entities in the domain's ontology which is usually large-scale, the state-space dimensionality becomes a factor of vital importance in a real-world SDS. Therefore, efficient approximation techniques must be incorporated to deal with this "curse of dimensionality".

- **Observation**: In POMDP, the agent receives a noisy observation $o \in O$ from the world. As identified by [128], the output of ASR and DST suffers from the different levels of noise and ambiguities, is what the agent gets and understands from the user.

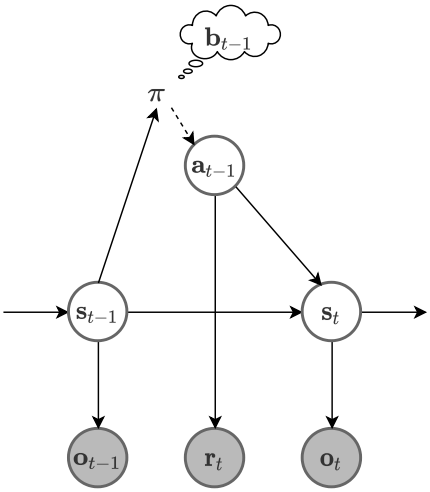


Figure 4.1 Graphical representation of POMDP. Shaded nodes denote the observable variables, whereas unshaded nodes represent variables that are not directly observable to the agent. The arrow with solid lines shows the direct influence, and the dashed line represents a distribution on the pointed variable. Variables a_t, s_t, o_t, r_t and b_t represent the *action*, *state*, *observation*, *reward* and the *belief-state*, respectively at time t.

- **Action**: Based on the current observation of the environment, the agent executes an action $a \in A$. It leads to a state transition which then updates the agent's understanding of the environment. For the statistical SDS paradigm, the action set is formed as the collection of all possible replies the system requires to make during the conversation. Due to the high variability in the language, a sentence is often denoted by a higher-level semantic representation, for example, a sentence, e.g. 'मैं शहर के केंद्रीय भाग में बंगाली खाना खोज रहा हूँ।' is represented by `inform(type=restaurant,food=बंगाली,area=केंद्र)`.

- **Transition probability**: In real-world applications, the dynamic nature of the environment causes uncertainty in the effect of each action. Due to this, the transitions between states are stochastic in general. Hence, a transition function $\mathcal{P}^a_{ss'} = P(s_{t+1} = s' | s_t = s, a_t = a)$ determines the probability of reaching to state s_{t+1} when the agent performs action a_t from state s_t.

- **Observation probability**: It is a probability $\mathcal{P}^a_{s'o'} = P(o_{t+1} = o' | s_{t+1} = s', a_t = a)$, where the agent observes o_{t+1} after executing action a_t and reaching state s_{t+1}. It is often considered as the accuracy of the system's sensing.

- **Reward**: A reward $\mathcal{R}^a_s = r(s_t = s, a_t = a)$ sets an objective that directs the agent to learn a desirable behaviour. It can be either stochastic or deterministic. Usually, it is a measure of how successful the dialogue was, e.g. whether the information that was asked by the user has been given and how efficient it was, e.g. how long the dialogue took [125].

- **Discount factor**: $\gamma \in [0, 1]$ determines the present value of the future rewards. Referring to Equation 4.1, a reward collected at k time steps in the future is worth only γ^{k-1} times than it would have received

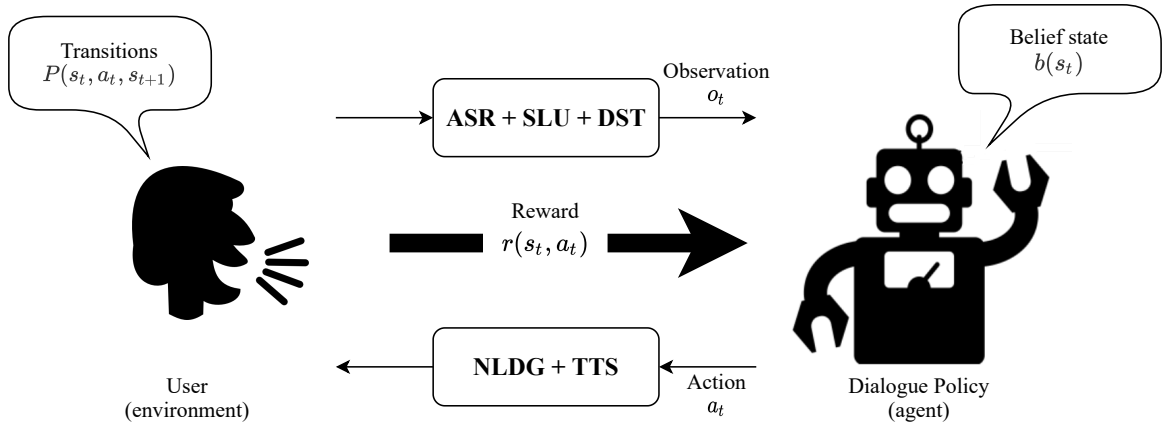


Figure 4.2 A loop of dialogue policy optimisation in the RL framework. Transitions $P(s_t, a_t, s_{t+1})$ are the probabilities of moving from state s_t to s_{t+1} on agent's last action a_t. o_t is the observation perceived by the agent. $r(s_t, a_t)$ denotes the reward received by the user at time t.

immediately. In an SDS, it is set close to one as the dialogue length is finite and, each dialogue turn is assumed to be equally important.

A schematic diagram of dialogue policy optimisation in an RL loop is illustrated in Figure 4.2. At each time step t, the agent receives an observation o_t from the user (environment), which is estimated by the ASR, SLU and DST components. The previous state s_{t-1} and the observation o_t will help infer the current state s_t. As the states are unobservable, the agent (dialogue policy) maintains a distribution over all possible states, called *belief state* $b(s_t)$. if the state space is S, the belief space can be represented as $[0, 1]^{|S|}$. For example, the initial distribution over all states will be shown as $\mathbf{b}_0 = [b(s_0 = s^1), ..., b(s_0 = s^{|S|})]^T$. Based on the currently estimated belief $b(s_t)$, the agent selects an action a_t which is then converted to natural speech using Natural Language Dialogue Generation (NLDG) and Text-To-Speech (TTS) components. Consequently, on performing action a_t, the agent collects a reward $r(s_t, a_t)$. It causes the user (environment) to transition to state s_{t+1} with probability $P(s_{t+1} = s' | s_t = s, a_t = a)$.

The dialogue management in the POMDP framework not only learns to update the belief over dialogue state but also determines a good policy [187]. Recently, these two tasks have been decomposed into a DST and a dialogue policy component, which achieved better overall performance. The DST in the Hindi domain is intensively studied in [1]. The focus of this work is concerned with learning a dialogue policy: a function $\pi(\mathbf{b}(s)) = a$ that determine which action to take given a belief distribution $\mathbf{b}(s)$ of the current dialogue state.

4.3 Overview of Dialogue Policy Optimisation Methods

The spoken dialogues is an episodic RL task as it is considered to follow a finite number of steps T (typically in the range of 2 to 20). The objective of the dialogue policy π is to maximise the cumulative discounted turn-based reward collected over the entire dialogue, as shown in Equation 4.1.

Table 4.1 Comparison of the RL approaches: whether they learn the value-function, the policy or both.

RL methods	Value function	Policy gradient function
Value-based	✓	-
Policy-based	-	✓
Actor-critic	✓	✓

In general, there are two categories of methods: *value-based* and *policy-based* methods that are usually adapted to estimate an optimal policy π^* . Both are distinguished in Table 4.1, where the intersection of the two is often referred to as the *Actor-Critic* method, which is the main attention of this chapter. Mathematical descriptions of each approach are elaborated in the following sections.

Since the SDS task assumes the input utterances as a set of finite and discrete semantics (states), a POMDP with finite-state is therefore adopted. Considering a discrete-state POMDP as a continuous-state MDP, the policy optimisation methods in the following sections are primarily explained in the context of MDPs as they can easily be extended to infer POMDPs.

4.3.1 Value-based Methods

When modelling the MDPs with value-based approaches, the expected discounted reward R_t^π at state $s \in S$ following the policy π is often determined by the *value function* $V^\pi : S \rightarrow \mathbb{R}$:

$$V^\pi(s) = E_\pi(R_t^\pi | s_t = s) = E_\pi\left(\sum_{i=0}^{\infty} \gamma^i r_{t+i+1} | s_t = s\right) \tag{4.2}$$

where, expected accumulated reward E_π is calculated over all possible state sequences generated by the policy π starting with the current state s_t.

The value function can not be used only to estimate the expected discounted reward on states but also on actions. In the same way, the *Q-function* $Q^\pi : S \times A \rightarrow \mathbb{R}$ is therefore defined as the expected discounted reward that is collected when action a is taken in state s following the policy π:

$$Q^\pi(s,a) = E_\pi(R_t^\pi | s_t = s, a_t = a) = E_\pi\left(\sum_{i=0}^{\infty} \gamma^i r_{t+i+1} | s_t = s, a_t = a\right) \tag{4.3}$$

where, the expected E_π is computed over all possible state-action sequences that can be generated with policy π.

From Equations 4.2 and 4.3, the value function $V^\pi(s)$ and the Q-function $Q^\pi(s,a)$ can hold the following relation with each other:

$$V^\pi(s) = Q^\pi(s, \pi(s)) \tag{4.4}$$

The reinforcement learning objective is to obtain an optimal policy, i.e. the policy that maximises the Value function. Assuming a finite state space S, the optimal Value function[2] can be solved using *Bellman optimality equation* $V^* = BV^*$ [240], where B is the Bellman operator:

$$V^*(s) = \max_a \sum_{s \in S} \mathcal{P}^a_{ss'} \cdot \left(\mathcal{R}^a_s + \gamma V^*(s')\right) \tag{4.5}$$

In a similar way, the optimal Q-function $Q*(s,a)$ is expressed by:

$$Q^*(s,a) = \sum_{s \in S} \mathcal{P}^a_{ss'} \cdot \left(\mathcal{R}^a_s + \gamma \max_{a'} Q^*(s',a')\right) \tag{4.6}$$

The optimal value function $V^*(s)$ and the optimal Q-function $Q^*(s,a)$ are related by:

$$V^*(s) = \max_a Q^*(s,a) \tag{4.7}$$

Hence, the optimal policy π^* can be implicitly derived either from the optimal value function:

$$\pi^*(s) = \arg\max_a \sum_{s' \in S} \mathcal{P}^a_{ss'} \cdot \left(\mathcal{R}^a_s + \gamma V^*(s')\right) \tag{4.8}$$

or from the optimal Q-function:

$$\pi^*(s) = \arg\max_a Q^*(s,a) \tag{4.9}$$

with selecting an action that maximises the corresponding value function or Q-function.

On the other hand, solving the POMDP is harder to model since the true state is not observable. It consists of a policy $\pi : B \rightarrow A$ that determines the best action a to be taken at each belief state $\mathbf{b}$ by maximising the expected total discounted reward. However, the true state is unknown, the optimal value function for a state can still be computed using the transition probability $\mathcal{P}^a_{ss'}$ and observation probability $\mathcal{P}^a_{s'o'}$ [237] as follows:

$$V^*(s) = \max_a \sum_{s' \in S} \mathcal{P}^a_{ss'} \cdot \left(\mathcal{R}^a_s + \sum_{o' \in O} \mathcal{P}^a_{s'o'} \gamma V(s')\right) \tag{4.10}$$

In the same way, the optimal Q-function for POMDPs can also be computed as follows:

$$Q^*(s,a) = \sum_{s' \in S} \mathcal{P}^a_{ss'} \cdot \left(\mathcal{R}^a_s + \max_{a'} \sum_{o' \in O} \mathcal{P}^a_{s'o'} \gamma Q(s',a')\right) \tag{4.11}$$

[2]Strictly, the optimal Value function should be denoted as $V^\pi(s)$, where π^* is the optimal policy. In order to keep the notation simple, the policy is denoted as V^*

Then, the optimal value function and Q-function for any possible belief state **b** can be computed using the respective $V^*(s)$ and $Q^*(s,a)$ values:

$$V^*(\mathbf{b}) = \sum_{s \in S} b(s_t = s) V^*(s) \tag{4.12}$$

$$Q^*(\mathbf{b}, a) = \sum_{s \in S} b(s_t = s) Q^*(s, a) \tag{4.13}$$

where $b(s_t = s)$ is the value of the belief state for the state s at turn t.

Due to the continuity of the belief state, it is difficult to find the exact algorithm for POMDP. In a real-world task, solving a POMDP thus requires approximations methods [241]. Point-based methods [242, 243], however, are found to be effective for modelling relevant, reachable belief states, but they are unable to scale to sizeable state-action space and are often model-based approaches that require the estimation of the environment (transitions). Model-free RL methods are another family for solving POMDPs that can directly update the value functions by appropriately exploring the environment and exploiting the learnt policy. For such methods, sample-efficiency is an essential characteristic for whether the model can realistically be employed on-line for live applications. Gaussian-Process State-Action-Reward-State-Action (GP-SARSA) [124] and Temporal-Difference (TD) [123] are methods that can estimate the sparse value functions from minimal numbers of training samples in model-free scenarios and are thus helpful for real-time SDS.

4.3.2 Policy-based Methods

In place of defining by value functions (V, Q), the policy π can also be directly parametrised. The methods are categorised to policy-based methods, or *Policy Search* [244] that directly operate in the parameter space θ to obtain the policy $\pi_\theta(a|s)$ (or $\pi_\theta(a|b)$ in POMDP) with parameters $\theta \in \Theta$ instead of learning a value function. It is distinguished in Table 4.1.

Recalling Equations 4.1 and 4.3, the policy π^* of an episodic RL task can be represented as the optimisation problem to maximise the expected reward $J(\pi)$:

$$\pi^* = \max_{\pi} J(\pi) = \max_{\pi} E[R|\pi] \tag{4.14}$$

where, R is the total episodic reward collected using policy π. if the policy is parametrised on parameter θ as π_θ, the objective is to obtain the optimal policy by maximising the expected reward $J(\theta)$:

$$\theta^* = \arg\max_{\theta} J(\theta) = \arg\max_{\theta} E[R|\pi_\theta] \tag{4.15}$$

As seen in value-based methods, the policy-based methods can also be divided into model-based and model-free approaches. The former fully understand the operating environment's dynamics (transition probability) while the latter does not. Hence, model-free approaches are needed to sample the environment during estimation. A set of model-free methods that find the optimal solution through a derivative-free optimisation approach considers the entire process as a black-box and uses evolutionary strategies algorithms for heuristic searching. They follow iterative modelling of learning the policy where at each iteration (generation), the current best estimate of parameters θ is perturbed (mutated) for generating a new population (next generation) which then evaluated based on the objective function, the total reward R. The parameter vectors with the highest scores on the objective function are then used in the next iteration to find the optimal policy. Some popular approaches, such as the cross entropy method [245] and natural evolution strategies [246, 247], have been successfully incorporated into many real-world problems. Due to the purely guessing method and agnostic to the given problem, these approaches suffer from inefficiency and incomprehensibility and therefore can not be applied to large-scale complex problems.

Policy-gradient methods, on the other hand, are a subclass of policy-based methods that estimate an optimal policy's weights through *gradient ascent* [248]. This assures an improvement and guarantee in finding the *local* optimum. The method learns the policy through a stochastic process. The system is going to select an action from the output probability distribution at each iteration. It means that if system observes the same dialogue state twice, it may not end up taking the same action twice. This improves the system to behave in a more natural way. With respect to the parameter θ, the policy gradient of $J(\theta)$ is derived as follows, assuming a trajectory τ sampled from policy π_θ:

$$
\begin{aligned}
\nabla_\theta J(\theta) &= \nabla_\theta E_{\tau\sim\pi_\theta}[R(\tau)] = \nabla_\theta \int_\tau p(\tau|\pi_\theta)R(\tau)d\tau \\
&= \int_\tau \nabla_\theta p(\tau|\pi_\theta)R(\tau)d\tau \\
&= \int_\tau p(\tau|\pi_\theta)\frac{\nabla_\theta p(\tau|\pi_\theta)}{p(\tau|\pi_\theta)}R(\tau)d\tau \\
&= \int_\tau p(\tau|\pi_\theta)\big[\nabla_\theta \log p(\tau|\pi_\theta)R(\tau)\big]d\tau \\
&= E_{\tau\sim\pi_\theta}\big[\nabla_\theta \log p(\tau|\pi_\theta)R(\tau)\big]
\end{aligned}
$$

The derivation estimates the simplified formula for policy gradient through the gradient-based expectation of total reward of a sampled trajectory τ. From third to fourth setup, the derivation follows the *Likelihood ratio trick* [249]: $\nabla \log p(x) = \frac{\nabla p(x)}{p(x)}$, where $\nabla \log p(x)$ is the *score function*. Further, the trajectory probability $p(\tau|\pi_\theta)$ can be expanded through the chain rule of the probability theory as follows:

$$p(\tau|\pi_\theta) = \mu(s_0)\pi(a_0|s_0,\theta)p(s_1,r_1|s_0,a_0)\pi(a_1|s_1,\theta)...\pi(a_{T-1}|s_{T-1},\theta)p(s_T,r_T|s_{T-1},a_{T-1})$$

where, $\mu(s_0)$ denotes the distribution of the initial state s_0. During the calculation of $\nabla_\theta \log p(\tau|\pi_\theta)$, the sequence of product converts into a sum, and the differentiation with respect to θ cancel out the terms $\mu(s_0)$ and $p(s_t, r_t|s_{t-1}, a_{t-1})$ which make it a type model-free RL approach (no need to know the dynamics (transition probability) of the environment). Hence, the final equation becomes as:

$$\nabla_\theta J(\theta) = E_{\tau\sim\pi_\theta}\left[\sum_{t=0}^{T-1} \nabla_\theta \log \pi(a_t|s_t) R(\tau)\right] \tag{4.16}$$

Based on the estimated gradient ∇_θ, the policy parameters θ is updated with a learning rate α as follows:

$$\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta) \tag{4.17}$$

To obtain a single reward at turn t in the trajectory, Equation 4.16 is supposed to be revised as:

$$\nabla_\theta E_{\tau\sim\pi_\theta}[r_t] = E_{\tau\sim\pi_\theta}\left[r_t \sum_{t'=0}^{t} \nabla_\theta \log \pi(a_{t'}|s_{t'})\right] \tag{4.18}$$

The total reward r_t is estimated on taking a sequence of actions $a_{t'}$, where $0 \leq t' \leq t$. Over the entire trajectory $\left(\sum_{t=0}^{T-1}\right)$, the accumulated reward will be calculated by:

$$\nabla_\theta E_{\tau\sim\pi_\theta}[R(\tau)] = E_{\tau\sim\pi_\theta}\left[\sum_{t=0}^{T-1} r_t \sum_{t'=0}^{t} \nabla_\theta \log \pi(a_{t'}|s_{t'})\right] \tag{4.19}$$

$$= E_{\tau\sim\pi_\theta}\left[\sum_{t=0}^{T-1} \nabla_\theta \log \pi(a_t|s_t) \sum_{t'=t}^{T-1} r_{t'}\right] \tag{4.20}$$

where, the term $\left(\sum_{t'=t}^{T-1} r_{t'}\right)$ in the above equation is the total reward collected from time-step $t \rightarrow T-1$. It can be replaced further by Q-value function from Equation 4.3:

$$\nabla_\theta E_{\tau\sim\pi_\theta} = E_{\tau\sim\pi_\theta}\left[\sum_{t=0}^{T-1} \nabla_\theta \log \pi(a_t|s_t) Q^{\pi_\theta}(s_t, a_t)\right] \tag{4.21}$$

This equation is called as *policy-gradient theorem* [248].

In practice, the gradient is estimated on a batch of N samples collected using the policy π_θ by interaction with the environment, where the accuracy increases as $N \rightarrow \infty$. One common way to solve this is the RL algorithm (Monte-Carlo method) [250], which directly calculates the gradient on total reward over all N samples following the policy π_θ:

$$\widehat{\nabla_\theta J(\theta)} = \frac{1}{N}\sum_{n=1}^{N}\left(\sum_{t=0}^{T-1}\nabla_\theta \log \pi(a_t|s_t)\sum_{t'=t}^{T-1} r_{t'}\right) \tag{4.22}$$

However, such gradient methods suffer from slow learning due to unable to handle high variance in the stochastic way of exploration. In addition to this, they are also sample-inefficient, which makes them unsuitable for a real-world problem. To solve this, a baseline function $b(s)$ is utilised to reduce the variance without changing the gradient [251]:

$$\widehat{\nabla_\theta J(\theta)} = \frac{1}{N}\sum_{n=1}^{N}\left(\sum_{t=0}^{T-1}\nabla_\theta \log \pi(a_t|s_t)\left(\sum_{t'=t}^{T-1} r_{t'} - b(s_t)\right)\right) \tag{4.23}$$

The difference term $\sum_{t'=t}^{T-1} r_{t'} - V(s_t)$ demonstrates whether the total reward using the current policy is better than *expected* $\big(V(s_t)\big)$ depicting how accurate the selection of current action a_t is. Here, the baseline function can be any arbitrary function. However, the best candidate for this baseline $b(s)$ is the value function $V(s)$ [252].

Another way to resolve the high variance issue, one can adopt a separate function, a *critic* with parameters w, to estimate the Q-function $Q^{\pi_\theta}(s_t, a_t)$ in Equation 4.21, where:

$$Q_w(s_t, a_t) \approx Q^{\pi_\theta}(s_t, a_t) \tag{4.24}$$

This leads to the *actor-critic* algorithm [253]:

$$\nabla_\theta J(\theta) = E_{\tau\sim\pi_\theta}\left[\sum_{t=0}^{T-1}\nabla_\theta \log \pi(a_t|s_t) Q_w(s_t, a_t)\right] \tag{4.25}$$

which consists of two sets of parameters: θ for the actor (policy) and w for the critic in such a way that the gradient ascent direction of the actor is suggested by the critic during the learning. The critic policy is assumed to be a Value- or a Q-function; hence, most approaches belonging to value-based RL can be utilised here. This is why the actor-critic-based RL algorithms are capable of acquiring the positive aspects of both value-based and policy-based methods. Nevertheless, this method has the advantage of directly modelling the policy, and the critic provides a good estimation of the expected total reward to reduce the variance.

4.3.3 Taxonomy of RL Approaches

In the previous section, we discussed two main methods used for obtaining the optimal policy. In comparison, policy-based approaches have converged in a better way than the value-based methods. This is because the latter often diverge when optimised through approximation functions since they optimise in value space and a slight modification during the value-estimation can lead to a significant change in the policy space [248]. In the

experiments, we have explored GP-SARSA [124], a value-based method, neural network-based methods (DRL), that are trained with policy-based and actor-critic methods. In addition, there is a taxonomy of RL approaches based on different dimensions and characteristics:

Episodic and Continuing task: There are two types of RL problems: 1) *Episodic task*: where the learning process is carried on a set of finite-length episodes, 2) *Continuing task*: where the learning process has only a single episode that continues indefinitely. In an episodic task, the initial and terminal states are clearly defined, which set the boundary for a starting and ending on an episode, e.g. task-oriented SDS, mazes and games. For continuing tasks, the examples are like balancing a cart pole, personal-assistant chatbot.

Model-based and Model-free: The difference between model-based and model-free methods is whether knowledge (dynamics) of the environment is known. In the POMDP scenario, this knowledge includes the transition and observation functions. Dynamic programming based value iteration or policy iteration [251] are the instances of model-based methods. GP-SARSA, DQN belongs to the model-free category because they learn directly with the interaction or from the given dataset.

Exploration and Exploitation: For the model-free category, the uncertainty in taking action often leads to *exploitation/exploration* dilemma: 1) **exploration**: choosing a non-optimal (random) action given the current policy in order to get more information about the environment to better optimise the policy further. 2) **exploitation**: prefers the selection of optimal action based on the current policy. It expects to set a trade-off between the exploration and exploitation of the current policy to achieve optimal policy in a lesser amount of time.

On-line and Off-line: An *on-line* method typically update the agent's policy incrementally on each sample in the interaction. MC [250] and GP-SARSA [124] approaches come into this category. On the other hand, *off-line* (or *batch*) RL methods first collects a batch of samples then learns the optimal policy. It is more sample-efficient since more information is provided in one single parameter update, e.g. DQN.

On-policy and Off-policy: Sometimes, training an RL algorithm uses another policy to generate the training dialogue (episodes), which is referred to as the *behaviour policy*. This is in contrast to the policy to be optimised, which is called the *target policy*. When the actions are drawn from the target policy during the training, the methods are known as *on-policy* methods. SARSA [251] is an example of on-policy RL methods. On the other hand, if actions are generated from the behaviour policy, such methods are called *off-policy* methods. For example, Q-Learning is an off-policy RL method, as it updates the target policy with the samples generated from the behaviour policy [254].

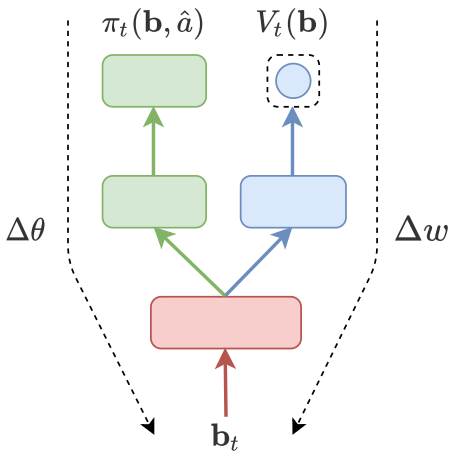


Figure 4.3 A2C architecture using feed-forward neural network, where $\mathbf{b}_t$ is Dialogue State at turn t, $\Delta\theta$ and Δw are the Policy and Value gradients, respectively. $\pi(\mathbf{b}, \hat{a})$ is predicted Policy Value of $\mathbf{b}$ some action $\hat{a}$, while $V(\mathbf{b})$ is corresponding Predicted Value function.

4.4 Proposed A2CER Method

As discussed in Section 4.3.2, for *Policy Gradient Theorem* [248] (see Equation 4.21), the objective function to learn the parameters of the gradient is:

$$\nabla_\theta J(\theta) = E_{\pi_\theta}\left[\nabla_\theta \log \pi_\theta(a|\mathbf{b}) Q^{\pi_\theta}(\mathbf{b}, a)\right] \tag{4.26}$$

Unfortunately, this form of gradient learning suffers from the problem of high variance. A baseline function is generally utilised to reduce the variance while not changing the estimated gradient [251]. However, the natural candidate for this baseline is the value function $V(\mathbf{b})$. Hence, Equation 4.26 get updates to:

$$\nabla_\theta J(\theta) = E_{\pi_\theta}\left[\nabla_\theta \log \pi_\theta(a|\mathbf{b}) A_w(\mathbf{b}, a)\right] \tag{4.27}$$

where, $A_w(\mathbf{b}, a)$ is the *advantage function* represented as $A_w(\mathbf{b}, a) = Q(\mathbf{b}, a) - V(\mathbf{b})$. It is a specialised version of *actor-critic*, where π_θ represents the *actor*, while $A_w(\mathbf{b}, a)$ denotes the *critic*. Hence, it is overall defined by two parameter sets, θ and w. Figure 4.3 depicts the architecture and model parameters of the obtained A2C policy.

In training A2C models, action selection is often made using a ϵ-greedy policy, which sets a trade-off between exploration and exploitation of the policy to either choose an action randomly with probability ϵ; or select on from the top-ranking actions. In such scenarios, the policy that is used to collect the training samples is called as a *behaviour policy* μ; on the other hand, the policy to be optimised is called *target policy* π. Hence, the A2C is an on-policy learning algorithm assuming that actions are drawn from the same policy as the target to be optimised ($\mu = \pi$).

4.4.1 A2C Experience Replay (A2CER)

Recently, several advancements, i.e. experience replay [233], off-policy retrace algorithm [255], have been applied in DRL to resolve various challenges in policy optimisation tasks. We introduced *experience replay* in the A2C method, whereby the model learns in an *off-policy* fashion. Typically, A2C policy learning is based on the on-policy RL algorithms where the training samples are recorded via the same policy which is being currently optimised. Such policy learning methods suffer from low sample efficiency. Experience Replay [255] is proposed to mitigate the issue whereby mini-batches of experiences (samples) are randomly selected from a *replay pool*, a kind of buffer with a pre-specified size for storing the previous experiences. It stabilises the learning process by reducing the data correlation, which is achieved by re-using the past samples in a series of updates. We encompass this ER based off-policy learning paradigm to A2C for training the dialogue policy.

In the A2CER policy, the dialogues sampled from old behaviour policy μ are used to update the current policy π. To correct the sampling bias generated by the old behaviour policy, we use an *Important Sampling (IS)* ratio [256] to rescale each estimated reward:

$$\rho_t = \min\left\{\frac{\pi(a_t|\mathbf{b}_t)}{\mu(a_t|\mathbf{b}_t)}, c\right\} \tag{4.28}$$

where, c is an upper limit constant, which is used to clip the IS weight to avoid potentially unbounded approximations. Thus Equation 4.27 (A2C) can be transformed by multiplying (rescaling) with IS ratio ρ as:

$$\nabla_\theta J(\theta) = E_{\pi_\theta}\left[\rho \nabla_\theta \log \pi_\theta(a|\mathbf{b}) A_w(\mathbf{b}, a)\right] \tag{4.29}$$

To reduce the number of parameters, we use an approximation over the advantage function by using the TD error [257] estimated by:

$$A_w(\mathbf{b}_t, a_t) = r_t + \gamma V_w(\mathbf{b}_{t+1}) - V_w(\mathbf{b}_t) \tag{4.30}$$

where, $A_w(\mathbf{b}_t, a_t)$ is estimated for each turn t using the current reward r_t and the difference between discounted future value-function $V_w(\mathbf{b}_{t+1})$ with γ and current value function $V_w(\mathbf{b}_t)$. w is parameters of behaviour policy μ. It substantially reduces the number of parameters required in estimating the critic A_w, in comparison to the simple advantage function used in the A2C method. For the A2CER, the off-policy for behaviour policy's π_μ parametrised value function V_w thus will be estimated by:

$$\nabla w^{\text{off}} = \sum_{t=0}^{T-1} \left(\bar{R}_t - \hat{V}_w(\mathbf{b}_t)\right) \nabla_w \hat{V}(\mathbf{b}_t) \prod_{i=0}^{t} \rho_i \tag{4.31}$$

where, $\bar{R}_t$ is the off-policy return estimated through Monte-Carlo return [258]. Finally, the gradient estimation for target policy π_θ is done by:

$$\nabla\theta^{\text{off}} = \sum_{t=0}^{T-1} \rho_t \nabla_\theta \log \pi_\theta(a|\mathbf{b}) \hat{\delta}_w \tag{4.32}$$

where, $\hat{\delta}_w = r_t + \gamma \hat{V}_w(\mathbf{b}_{t+1}) - \hat{V}_w(\mathbf{b}_t)$ is the TD error estimated using the value of $\hat{V}_w$.

Algorithm 1: A2CER Algorithm

Input: policy: $Q_\theta(\mathbf{b}, a)$, $\pi_w(a|\mathbf{b})$, hyperparameters: $batch\_size, \gamma, n, c$
Initialise θ, w and Q_θ(terminal)=0
repeat
  Generate an episode $\{\mathbf{b}_{0:T}, a_{0:T}, r_{0:T}\}$ through ϵ-greedy using $\pi_w(\cdot,\cdot)$
  Save generated episode in replay memory M
  for $i = 1$ *to* n **do**
    Sample a subset of M of size $batch\_size$
    for *each dialogue* $\{\boldsymbol{b}_{1:N}, a_{1:N}, r_{1:N}, \mu\}$ *in* M **do**
      $Q' = 0$
      for $t = N$ *to 1* **do**
        $\rho_t \leftarrow \min\left\{\frac{\pi_w(a_t|\mathbf{b}_t)}{\mu(a_t|\mathbf{b}_t)}, c\right\}$
        $Q' \leftarrow r_t + \gamma Q'$
        $V(\mathbf{b}_t) \leftarrow \sum_a Q_\theta(\mathbf{b}_t, a)\pi_w(a|\mathbf{b}_t)$
        $A'(\mathbf{b}_t, a_t) \leftarrow Q' - V(\mathbf{b}_t)$
        $A(\mathbf{b}_t, a_t) \leftarrow Q_\theta(\mathbf{b}_t, a_t) - V(\mathbf{b}_t)$
        $B \leftarrow \sum_a \rho_t \nabla_w \log \pi_w(a|\mathbf{b}_t) A(\mathbf{b}_t, a_t)$
        $g \leftarrow g + \rho_t \nabla_w \log \pi_w(a_t|\mathbf{b}_t) A'(\mathbf{b}_t, a_t) + B$
        $d\theta \leftarrow d\theta - \nabla_\theta (Q' - Q_\theta(\mathbf{b}_t, a_t))^2$
        $Q' \leftarrow \rho_t (Q' - Q_\theta(\mathbf{b}_t, a)) + V(\mathbf{b}_t)$
      end
      $w \leftarrow w + \alpha \cdot g$
      $\theta \leftarrow \theta + \alpha \cdot d\theta$
    end
  end
until *convergence*

We can also show that the A2CER is free from bias incurred from the baseline function. This is due to the fact that the baseline term becomes zero during the gradient estimate with the rule of constant integral probability distribution (see Appendix D for the proof). The above enhancements not only speed up the dialogue policy learning but also stabilise the training process in comparison with A2C. It is called Advantage Actor-Critic (A2C) with experience replay (A2CER).

The pseudocode of the training algorithm is presented in Algorithm 1. It performs ϵ-greedy exploration where selection of the optimal action is made through learned policy with 1-ϵ and a random action with probability ϵ. There are some hyperparameters used in the algorithm, i.e. $batch\_size$ controls the number of dialogues used under a training step, γ is a geometric discount factor, n decides the number of training steps for each new dialogue trajectory, and c is used to clip the IS weight. θ and w represent the parameters of the target policy

and behaviour policy which are updated during the training with corresponding gradients ∇_θ and ∇_w. For each dialogue, Q' stores the updated estimate of Q-function, which is determined based on state-action trajectories sampled from the experience replay memory. It helps estimate the current Q-function with off-policy interactions in a safe and efficient way. Similarly, $A(\mathbf{b}_t, a_t)$ and $A'(\mathbf{b}_t, a_t)$ denote the actual and updated advantage function; the actual one finds the possible action distribution while updated one used in estimating the off-policy gradient ∇_w. We investigate the effect of various hyperparameters, i.e. their tuning, in Section 4.6.2.

4.5 Experimental Environment

The Allahabad restaurant domain-based dialogue system is used for the experimental work described in this chapter. Users converse with the dialogue agent to find a restaurant matching their required constraints such as food, price range and area. The overall application-domain details, such as ontology and venue details, are mentioned in Section 2.1.2. Two operating modes are provided to perform the interaction: *live user trial mode* and *user simulation mode*. In live user trial mode, the user interacts with the system directly in a real-time environment. On the other hand, in user simulation mode, it uses an agenda-based simulated user to carry out the interaction with the system at the abstract dialogue act level [259]. It is helpful to generate a large amount of dialogue data covering a more comprehensive range of user-system interactions required to learn an efficient dialogue policy. The following section discusses the user simulator, reward estimation, dialogue evaluation and models to be evaluated in more detail.

4.5.1 User Simulator

When the dialogue policy is optimised with a reinforcement learning approach, either model-based or model-free optimisation algorithms can be applied (see Section 4.3). When exploring the model-based RL approach on an MDP model, the state transition probabilities (representing the dynamics of the environment), as defined in Section 4.2, are supposed to be available. On the other hand, for a POMDP, both transition and observation probabilities need to be known. These probabilities can be estimated on a given annotated dialogue corpus. However, it requires hundreds of thousands of dialogues to train a reasonable dialogue policy for a real-world task and faces sparsity issues requiring further approximations [260, 261].

Moreover, it is financially costly and time-consuming to collect such large amounts of dialogue data. In addition, the initial poor performance during the interactions may lead to a negative user experience. As a result, the model-free approach is often found to be more convenient, where the policy can be optimised directly in interaction with the real users. Hence, building a simulated user that can directly interact with the dialogue policy, would be very useful [262, 259].

$$\text{Goal} = \begin{bmatrix} \text{Constraint} = \begin{bmatrix} \texttt{food=बंगाली} \\ \texttt{price=सस्ता} \end{bmatrix} \\ \\ \text{Request} = \begin{bmatrix} \texttt{address=} \\ \texttt{phone=} \end{bmatrix} \end{bmatrix} \quad \text{Agenda} = \begin{bmatrix} \texttt{inform(food=बंगाली)} \\ \texttt{inform(price=सस्ता)} \\ \texttt{request(address)} \\ \texttt{request(phone)} \\ \texttt{bye()} \end{bmatrix}$$

Figure 4.4 An example of the user simulator task.

There are many techniques to build a user simulator, such as graph-based [263], agenda-based [232], and corpus-based [264–266, 259, 122] approaches. In principle, corpus-based statistical simulators are more desirable as they make the dialogue modelling process fully automatic [125]. These user simulators are expected to exhibit three key characteristics:

1. It should behave similar to a real rational user in a goal-directed scenario.
2. It must generate a coherent sequence of sentences/actions.
3. It should be able to generalise to new contexts [232].

The user simulator can interact with the system at either the dialogue act level, the word level or the speech level [267]. To effectively encode the dialogue history and user goal, *the agenda-based user simulator* is adopted in this thesis with estimated parameters as described in [122, 232]. In this method, the user state is factored into a *goal* and an *agenda*. The goal consists of a set of slot-value pairs representing the constraints and requests. On the other hand, the agenda stacks turn-level user intentions in semantic level (DA & slot-value pairs) form. An example of the user simulator is shown in Figure 4.4, where the left is the user goal, and the right represents the agenda. At the start of each dialogue, the user goal is randomly initialised with constraints, i.e. `food=बंगाली`, `price=सस्ता` and request attributes such as `address`, `phone`.

The agenda's role is to elicit the user's intentions in the dialogue acts form needed for the simulator to achieve the selected goal. It stores them in a stack-like structure, as shown in Figure 4.4. Depending on the situation, the stack pushes and pops a user's intention during the interaction. It ensures that the user simulator exhibits consistent, goal-directed behaviour across the entire conversation.

Both the goal and the agenda are dynamically updated throughout the dialogue based on specific *decision points*. The list of parameters representing the Friendly (standard user behaviour) and Unfriendly (user barely provide any extra information) distributions are given with their associated distribution (G for geometric and B for binomial) in Table 4.2. For example, taking up a situation, the simulator will relax its constraints when its initial goal can not be satisfied with the help of the parameter (ConstraintRelax). The Friendly simulator does not make the decision of removing a constraint from the user-goal with a probability of 0.667 for the parameter

Table 4.2 Parameter setting for the Friendly (std.) and Unfriendly agenda-based simulated users.

Parameter	Dist	Val. (Friendly)	Val. (Unfriendly)	Interpretation
InformCombination	G	0.600	0.100	Add a constraint to the goal
AddAttributeToReq	G	0.333	0.111	Add an attribute from the goal requests
YesAfterReqmore	B	0.250	0.600	Say yes without giving more info
AffirmWithAgdItem	B	0.050	0.050	When affirming, provide more info
Greeting	B	0.500	0.300	Respond to greeting
ConstraintRelax	B	0.667	0.333	Remove a constraint from the goal
TellAboutChange	B	0.500	0.200	Inform about a goal change
ByeOrStartOver	B	0.333	0.500	End the dialogue or start again
DealWithPending	B	0.500	0.300	Deal with pending items on the agenda
InformToConfirm	B	0.050	0.050	Change informs to confirms on agenda
AddEntityName	B	0.050	0.050	Provide entity name when requesting
NoAttrWithDontcare	B	0.800	0.900	Leave out attributes if their values do not matter
ReqAltsAfterEntRec1	B	0.143	0.143	Request alternative
ReqAltsAfterEntRec2	B	0.143	0.143	Request alternative and change goal
RequestResponce1	B	0.200	0.200	Repeat a random constraint
RequestResponce2	B	0.200	0.600	Make up a new constraint
CorrectingAct1	B	0.450	0.200	Correct a misunderstanding with negate
CorrectingAct2	B	0.400	0.100	Correct a misunderstanding with deny
OverruleCorrection	B	0.100	0.700	Do not correct a misunderstanding
ThankAck1	B	0.100	0.600	Say thank you
ThankAck2	B	0.100	0.300	Say ok

(ConstraintRelax) at each turn, while for the Unfriendly simulator, its (ConstraintRelax) value is set to 0.333. This decision point is only reached at particular stages of the dialogue.

The parameters (decision points) are estimated either as a deterministic set or a stochastic process [122]. The deterministic decision points are manually determined to preserve rational user behaviour. In contrast, the stochastic decision points are controlled by corpus-based parametrised probability distributions to enable variability in simulated user behaviour. To achieve that, a sample-based maximum likelihood technique is applied, where the simulator is run repeatedly for the given system acts in the corpus, and then the number of arbitrary decisions that lead to simulated acts matching true acts in the corpus are counted. Based on the counts for each random decision points, the parameters are then estimated. The agenda-based user simulator with parameters (both friendly and unfriendly) estimated from data as described in [122, 268], is used in current work.

Another critical component in the user simulator is the error model that has to comply with the noisy real-world interaction between the system and the user [232]. This model is used to add confusion in the user simulator output before it is passed to the dialogue manager, considering that the dialogue manager is not aware of the true state the user wanted to communicate but only receives a noisy version. It resembles the real-time interaction where the input revived from the speech understanding component is often corrupted with noise. Typically, the error model generates an N-best list of noisy dialogue acts by two types of models; 1) Uniform error model and 2) Dirichlet error model [269]. In the current work, we utilise the *uniform error model*, which

uses a fixed error rate to confuse each dialogue action in the N-best. Otherwise, the original dialogue act is considered to be in the N-best list.

Using the error model, the user simulator includes an error generator, which confuses the user inputs with different error rates at the semantic level, known as Semantic Error Rates (SERs) [270]. Based on this error rate, the actual act that will be output is computed randomly with a certain probability. In our Restaurant domain simulator, the user-act is confused with probability 0.2, while among slot-value pairs, slot and values are separately confused with probability of 0.3 and 0.5, respectively. Note that while these probabilities are fixed, they only decide how an individual dialogue-act is going to be confused. The overall amount of dialogues affected by the errors can still be changed by altering the SERs. In our work, we utilise three levels of semantic errors, i.e. 0%, 15% and 30% during the experiments.

4.5.2 Dialogue Evaluation and Reward Estimation

The evaluation of spoken dialogue systems is difficult due to the complexity of its long-term interaction between the user and the system. In contrast to most data-driven tasks in speech and natural language processing, for which the evaluation metrics are well-established [271, 272, 61], the definition of a good SDS is very vague. An SDS is built of distinct modules such as SLU and DST. Although there are defined evaluation methods for most of them, the joint evaluation of the whole system is still challenging. Evaluating the performance of the dialogue manager is itself hard due to the vast space of possible dialogues.

The most natural way of the evaluation is to have the dialogue manager interact with humans and let the human judges rate the interactions [273]. However, it is often infeasible to evaluate all possible dialogue due to its high cost and time-consuming nature. Nevertheless, different users may have their own subjective views of the 'goodness' of the dialogue. Thus, an alternative evaluation metric is desired, which uses the reinforcement learning reward function [125] to avoid the costly human-rating process.

In data-driven systems, a corpus of dialogue data is required to be collected and used to build supervised learning-based dialogue systems [201, 206]. The goal of such systems is to *mimic* the responses present in the data and be evaluated by similarity metrics, i.e. BLEU [274] and METEOR [275], which are widely used in machine translation. Despite that, it has been shown that these word-based similarity metrics exhibit a low correlation with human ratings [207]. In addition, the supervised training data often lack sufficient diversity and coverage of salient dialogue flows.

As the main focus of the thesis is task-oriented SDS, completing the task is a straight-forward automatic measurement of whether the information provided by the system matches the user goal. Furthermore, the dialogue quality is highly dependent on the performance of the dialogue policy, which manages and controls the flow of the dialogue. Nevertheless, this task-completion information can become a high-level learning objective

of the dialogue policy in the SDS pipeline. For RL based dialogue policy, it is often referred to as the *reward* function.

The reward function plays an important and critical role in the effective POMDP framework and achieving success in the RL task. It defines and models the desired behaviour of the learning agent [276]. A positive reward is typically given to the agent on reaching certain situations related to task success and vice versa. In several examples to task-oriented systems [277, 273, 128], the definition of success is often formulated based on a predefined task given to the users and estimated on whether all slots (user goals) mentioned by the user are fulfilled. In addition, a per turn penalty (negative reward) is applied to keep dialogue short.

The work presented in this thesis will use a simple reward function that encourages short and successful dialogues, which is effectively used in other studies [11, 187]. The reward function is defined in the following way. A dialogue is considered to be a success or failure based on the criteria of whether all of the users' constraints expressed during the conversations are completely satisfied or not. A positive reward of +20 will be given on the successful completion of the dialogue. Successful completion means that the system has provided the user an *entity* (*venue*) matching all the users' constraints and delivered all the information requested by the user about that particular entity. As users generally prefer shorter interactions, the reward function is designed to give a negative reward of -1 at each turn to encourage brevity. Hence, the cumulative reward for an entire dialogue episode is calculated as follows:

$$\mathcal{R} = 20 \times \mathbb{1}(\mathcal{D}) - N \tag{4.33}$$

where, N is total number of turns in the dialogue and $\mathbb{1}(\mathcal{D})$ is the indicator function to denote the success of a dialogue.

Table 4.3 The list of summary system actions.

System Action	**Description**
hello	system greets with hello
`request_area`	system asks user to provide the value of area
`request_pricerange`	system asks user to provide the value of price range
`request_food`	system asks user to provide the value of food type
`confirm_area`	system asks user to confirm the value of area
`confirm_pricerange`	system asks user to confirm the value of price range
`confirm_food`	system asks user to confirm the value of food
`select_area`	system asks user to choose from two values of area
`select_pricerange`	system asks user to choose from two values of price range
`select_food`	system asks user to choose from two values of food
`inform_system`	informs the slot-values with probability > threshold
`inform_byname`	system informs the name of an entity
`inform_alternatives`	system informs an alternative entity
`inform_requested`	system informs the value of the user-requested slots
`bye`	system says good bye

Sys : hello() → hello()

User : inform(type=restaurant)

Sys : request_area() → request_area()

User : inform(area=केंद्र)

Sys : inform() → inform(name="हरि वृंदावन", food="शाकाहारी")

User : reqalts()

Sys : inform() → inform(name="बीजिंग स्ट्रीट", food="चाइनीस")

User : request(address)

Sys : inform_requested() → inform(name="बीजिंग स्ट्रीट", address="मेन रोड, रामबाग, इलाहाबाद")

User : thankyou()

Sys : bye() → bye()

Figure 4.5 An example dialogue elucidating the relationship between the summary and master actions. The interactions are represented in a semantic dialogue-act form, where the system responses are written as "Sys: summary_action → master_action".

4.5.3 Action Spaces

Action space represents the set of *system actions* (dialogue acts) that the system can give as a response. This is also called *master action space*. In the case of Allahabad restaurant domain (see Appendix A), there are four informable slots of an entity, each with a binary choice of whether the system inform about it. A single inform action, thus, makes up $2^{(4)}$=16 separate master actions, distinguished only by what they inform about. Similarly, the requestable slots (7) form 128 different master actions. Adding up the other acts, i.e. confirm, select, leads to a significant set of possible master actions.

Due to its large size, training a dialogue policy in this action space is complex and error-prone. Some RL algorithms are unable to converge to the optimal policy or take too much time to converge. Nevertheless, they also have to deal with the *prohibitive computation demands*. For instance, if training is done in on-line mode, the user may have to wait a significant amount of time for the system to reply. To alleviate this problem, we use a fixed set of *summary actions* reduced to a much less number of actions than the master actions. The summary action space consists of a set of slot-dependent, e.g. `request_pricerange`, `confirm_area`, and slot-independent, e.g. `hello`, `bye`, summary actions, as shown in Table 4.3. If the dialogue policy is trained on this condensed action set, the selected action must be converted to *master action*. An example dialogue is shown in Figure 4.5, between a user looking for a restaurant in centre (केंद्र) part of the town, and a system that implicitly translates between summary and master actions.

Table 4.4 Statistical details of the HDRS corpus [1].

Description	Values
Total #Dialogues	1400
Total #Turns	5763
Avg Turns per dialogue	4.12
Avg Tokens per user-utterance	8.41
Avg Tokens per system-utterance	12.20
#Dialogues with goal change	557

It is also observed from Table 4.3 that summary actions require slots or slot-value pairs to reconstruct the system act. The grounding information is used to identify these slot-value pairs [278]. For example, if the summary action is `inform_requested`, then all the slot-value pairs requested by the user are added to form the master system act. It would help in assuring the user that offered venue has the requested properties. Thus, the conversion is done heuristically with a set of hand-crafted rules that map each summary action to a master action by finding the optimal slots to inform on the given belief state.

4.5.4 Pre-learning from Demonstration Data

Training dialogue policy on-line with the RL methods often suffers from the cold start issue. In early-stage learning, the RL approaches do not have long-term planning capability, leading to unacceptable behaviour from the user's perspective. To alleviate the problem, an offline *demonstration data* is utilised to bootstrap the policy. This is similar to the training procedure recently adopted by many game playing applications [279, 280]. Such data may be collected from a WOZ paradigm [1] or can be obtained from the interaction between the users and an existing policy.

In this work, we utilise the Hindi Dialogue Restaurant Search corpus [1] as a demonstration data for the pre-training. The corpus consists of 1400 human-human conversations on the restaurant domain collected using the Wizard-of-Oz paradigm. The dialogues are system-initiated, where each turn contains a pair of system and user utterances, a belief state, and a set of informed or requested slot-values currently mentioned by the user. Table 4.4 presents the statistical details of the corpus.

The pre-training has the objective to *mimic* the response behaviour from the corpus. It is essentially performed as Supervised Learning (SL) procedure. Like in RL algorithms, the input to the model is the belief state $\mathbf{b}$ and output is the system action a, and the training objective is to minimise a *joint cross-entropy loss* $\mathcal{L}(\theta) = -\sum_k y_k \log(p_k)$ between action labels y and model predictions p for each sample k, where θ represents the parameter set of the training policy. Thus, the single model is trained using both SL and RL with different training objectives without modifying the architecture. Another way, the demonstration data can be used to initialise a supervised replay buffer to enhance the early-stage performance or with a combination of both.

Pre-training the policy by SL on a fixed corpus may not generalise well. This is because the noise levels in spoken dialogues may vary across conditions, which significantly affect the performance. In addition, a policy trained using SL can not perform long-term planning, an essential property for a robust and natural dialogue manager. However, this supervised pre-training offers a good dialogue policy at an initial stage that can further be fine-tuned either by simulated or real-time user interactions using RL.

4.5.5 Model Comparison

All the experiments in this chapter utilised the software PyDial toolkit [191], which is developed as a framework for modular SDS. All models are given a full dialogue belief state **b** of size 272 as input, including the last system action, distribution over the user intention, the informable slot-values and the requestable slots. The output consists of 15 labels denoting a summary action space that determines the system intent at the semantic level, as shown in Table 4.3. We have utilised two value-based methods, i.e. GP-SARSA [124], DQN [129] and a hybrid of policy-based (*target policy*) and value-based (*critic policy*) model that is Advantage Actor-Critic (A2C) [130] to compare with the proposed A2CER method for policy learning. The characteristics of the methods are compared in Table 4.5.

Table 4.5 Overview of the RL models used for learning the dialogue policy.

Model type	**GP-SARSA**	**DQN**	**A2C**	**A2CER**
	non-parametric value-based	**parametric value-based**	**parametric policy-based**	**parametric policy-ER-based**
Value function	✓	✓	✓	✓
Policy function	-	-	✓	✓
Experience replay	-	✓	-	✓
Train by backpropagation	-	✓	✓	✓
Computational complexity	cubic*	linear	linear	linear

† *In the size of a set of representative points subjective ratings [124].

GP-SARSA

GP-SARSA is a state-of-the-art model-free and value-based RL algorithm that has been proven effective for dialogue policy learning [124]. As a value-based method, it learns the dialogue policy by a Q-function which is modelled by the Gaussian Process (GP) $\mathcal{GP}(m(\cdot,\cdot), k(\cdot,\cdot))$:

$$Q(\mathbf{b}, a) \sim \mathcal{GP}(m(\mathbf{b}, a), k((\mathbf{b}, a), (\mathbf{b}, a))) \tag{4.34}$$

where, $m(\cdot,\cdot)$ is the prior mean function, and $k(\cdot,\cdot)$ is the kernel function, which is factored into separate kernels over belief and action spaces as $k_{\mathcal{B}}(\mathbf{b},\mathbf{b}')k_{\mathcal{A}}(a,a')$. Using the idea of TD for greedy policy estimation of Q-function, the SARSA algorithm is applied, which iteratively updates the Q-function on-line using the rule:

$$Q(\mathbf{b},a) \leftarrow Q(\mathbf{b},a) + \alpha\big[r(\mathbf{b},a) + \gamma Q(\mathbf{b}',a') - Q(\mathbf{b},a)\big] \tag{4.35}$$

here, $\mathbf{b}'$ and a' are the next belief-state and next action in the dialogue trajectory. When the Q-function correction provided by the right-hand side rule of Equation 4.35 is used to estimate the posterior in GP-based Q-function, the GP-SARSA algorithm is obtained [124] based on the triplets of belief-action pairs $(\mathbf{b},a)$ and their corresponding rewards.

GP-based RL (GP-SARSA) is observed to be a promising algorithm as it can learn from a small sample of observations. It is better at exploiting the correlations defined by a kernel function and providing an uncertainty measure of its estimates. As an approximation method, this knowledge of the distance between data points in the observation space greatly speeds up the policy learning because the Q-values of the unexplored space can be estimated from the Q-values of nearby points. To avoid the burden of memorising every data point, which makes the computation and model complexity intractable, sparse approximation methods, i.e. *kernel span* [281], are used to reduce the size of stored training points.

Deep Q-Network (DQN)

The GP-SARSA has provided an estimate of the uncertainty with an underlying approximation function that helps not only to deliver sample-efficient policy learning but also handle the ASR/SLU errors. But, the use of the sparse approximations tricks (kernel-span algorithm) restricts it to be utilised in very large training sets, hence becoming unsuitable to be used for commercial wide-domain SDS.

On the other hand, the recent introduction of deep RL methods [282, 129, 280] has shown a significant potential for dialogue policy optimisation due to their high flexibility and scalability. We utilise the *Deep Q-Network*, a neural-network based variant of the Q-learning algorithm, to approximate the Q-function for optimal dialogue policy. As it is also based on a model-free way of RL algorithm, the optimal policy π^* is learned through the modified version of Q-function (given in Equation 4.11) of *Bellman equation* [240] as:

$$Q(\mathbf{b},a) = E_{\pi^*}\big\{r(\mathbf{b},a) + \gamma \max_{a'} Q(\mathbf{b}',a')|\mathbf{b},a\big\} \tag{4.36}$$

It is based on the sequential approximation, where the loss is minimised by:

$$L(w) = E\big[(y - Q(\mathbf{b},a;w))^2\big] \tag{4.37}$$

where, $y = r + \gamma \max_{a'} Q(\mathbf{b}', a'; \bar{w})$ is the target value to update the parameters w taking its gradient under the category of *off-policy* methods. Note that the target value y is estimated by a *target network* $\bar{w}$ which is updated less frequently than the *main network* w. It helps stabilise learning by avoiding the high correlation situation among samples of belief states and actions, hence achieving better performance.

Advantage Actor-Critic (A2C)

Although DQN has resolved the scalability issues of the dialogue policy learning with high convergence capability, they are typically slow gradient-based methods due to low sample efficiency, which is problematic for on-line learning with real users. To speed up the policy learning with better performance, an actor-critic type of policy method is utilised. The A2C is an on-policy learning algorithm assuming that actions are drawn from the same policy as the target to be optimised ($\mu = \pi$). The details are given in Section 4.4.

A2C Experience Replay (A2CER)

Typically, A2C policy learning is based on the on-policy RL algorithms. Due to this, it suffers from low sample efficiency. As explained in Section 4.4.1, Experience-Replay (ER) [255] is introduced to mitigate the issue whereby mini-batches of experiences (samples) are randomly selected from a *replay pool*, a kind of buffer with a pre-specified size for storing the previous experiences. It stabilises the learning process by reducing the data correlation, which is achieved by re-using the past samples in a series of updates. Here, ER is the collection of past dialogue experiences. The past experiences are collected from different policies rather than the current policy, which is being optimised. The use of ER leads to *off-policy* updates.

In the A2CER policy, the dialogues sampled from old behaviour policy μ are used to update the current policy π. To correct for the sampling bias (generated by the old behaviour policy), we use an *Important Sampling (IS)* ratio [256] to rescale each estimated reward. The details are given in Section 4.4.1. The above enhancements not only speed up the dialogue policy learning but also stabilise the training process in comparison with A2C. It is called Advantage Actor-Critic (A2C) with experience replay (A2CER).

Rule-Based Policy

In addition to the data-driven RL algorithms, we have also evaluated the performance of handcrafted rule-based policy under various environment settings. The actions for the policy are carefully designed based on the heuristics based on the corresponding belief state [283].

4.6 Results and Discussion

In this section, we evaluate the performance of A2CER adapted for the Hindi language spoken dialogue system. The models discussed above are first evaluated based on the 0% SER embedded within the agenda-based user simulator, which helps construct the user response in the semantic form (dialogue-act) [259]. It represents the dialogue scenarios where the user input is perfectly caught and tracked in the dialogue belief state without noise. For simulating a more challenging environment, both standard and unfriendly simulated users are considered for comparison on various SERs 0%, 15% and 30%. Next, we investigate the effect of using *demonstration data* in mitigating the cold start problem and the performance of RL models on comparing *master and summary action space*. It is evident that the A2CER exhibits comparative performance and fast convergence over the other models.

Normalised to unit, the total return of each dialogue is set to $\mathbb{1}(\mathcal{D}) - 0.05 \times N$ (see Section 4.5.2), where N is the dialogue length and $\mathbb{1}(\mathcal{D})$ is the success indicator for dialogue $(\mathcal{D})$. We set the maximum dialogue length to 25 turns and the discount factor γ to 0.99. The evaluation metrics are the *average success rate* and *average reward* for comparing policy models. The success rate is defined by the percentage of dialogues where the dialogue manager has successfully fulfilled the user goal by providing the desired restaurant venue details. At the same time, the average reward is estimated by the sum-averaged of the final reward collected at the end of each dialogue.

4.6.1 Reinforcement Learning from Scratch

The models are configured to perform dialogue modelling with a specific setting. GP-SARSA uses a *linear kernel* to represent the state space and a *delta kernel* for the action space. All the deep RL models (DQN, A2C and A2CER) contain two hidden layers with the size of 130 and 50 neurons, respectively. The Adam optimiser [284] with a 0.001 learning rate α is used to optimise the model parameters. An ϵ-greedy policy is utilised to establish the exploration/exploitation trade-off initially set to 0.3 and iteratively reduced to 0.0 over 4000 training dialogue samples. It helps to prefer exploration at the initial stage and exploitation at the end of the training duration. In contrast, GP-SARSA handles this trade-off automatically.

Figure 4.6 shows the learning curves of success rate, rewards and number of turns, respectively, for the dialogue policy optimised with GP-SARSA, DQN, A2C and A2CER methods. After every 200 training dialogues, all the models are tested with 600 dialogues. Previous research shows that the GP-SARSA learns very fast and is comparatively stable under smaller application-domain task-oriented dialogue. However, A2C and A2CER performance are also comparable to others. DQN, on the other hand, is highly unstable as it learns the parameters only for the Q-function, which suffers from the issues of high-variance and low sample-efficiency. It proves that an iterative improvement in value space does not guarantee an improvement in policy space. The

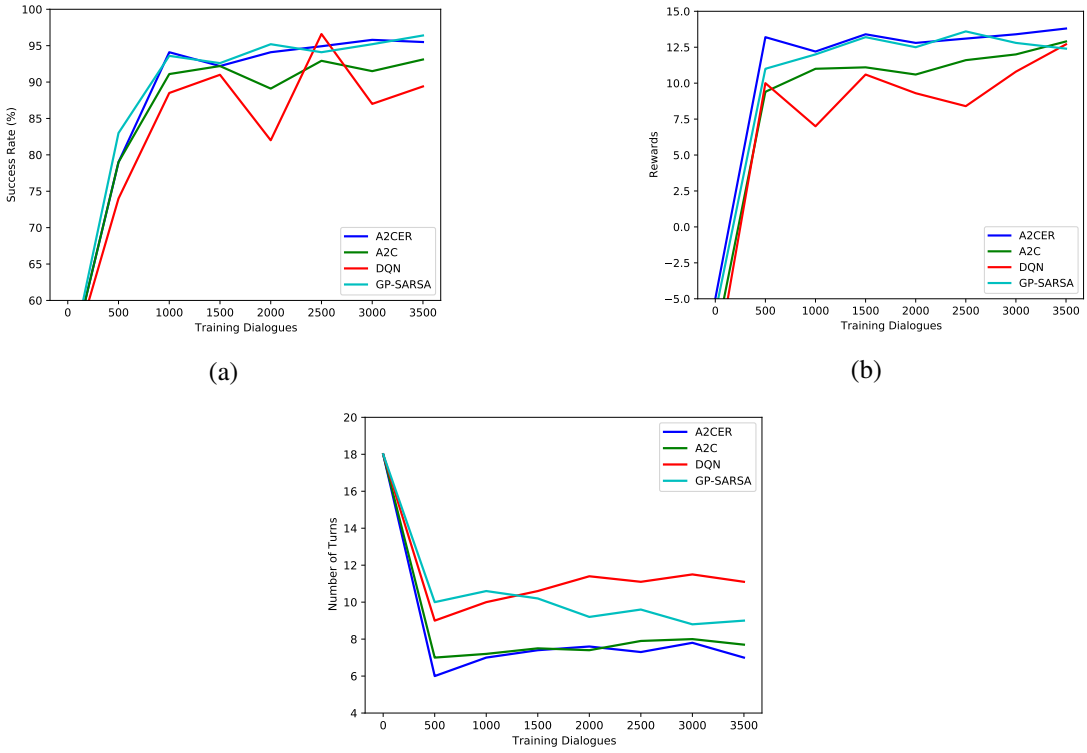


Figure 4.6 Comparison of A2CER to other RL methods, i.e. GP-SARSA, DQN and A2C, with user simulation under noise-free conditions on (a) Success rate, (b) Rewards and (c) Number of turns.

core of A2C and A2CER models is that it enhances stability by learning parameters separately for both the policy (actor) as well as value-function (critic).

Furthermore, it is also evident that the inclusion of *experience replay* improves the A2C performance significantly. Learning with the help of experience replay, A2CER achieves high sample efficiency, which reflects in the performance comparison with A2C. Thus, the A2CER has not only the capability of stable learning by reducing the variance as an actor-critic method but is also better at handling low sample-efficiency as an ER-based RL method.

Table 4.6 Reward and success rates of the four policy models with Standard and Unfriendly user simulator under three different values of SER, i.e. 0%, 15% and 30%. The highest reward obtained by a data-driven model in each row is highlighted. (Suc.= Success rates (average), Rew.= Reward (average))

User	**SER(%)**	**GP-SARSA**		**DQN**		**A2C**		**A2CER**		**Rule-Based**	
		Suc.	**Rew.**	**Suc.**	**Rew.**	**Suc.**	**Rew.**	**Suc.**	**Rew.**	**Suc.**	**Rew.**
Standard	0%	98.4%	12.4	89.4%	12.7	93.1%	12.9	95.5%	**13.8**	100.0%	15.0
	15%	96.6%	**12.8**	84.8%	11.6	90.8%	12.3	94.6%	12.5	98.8%	13.4
	30%	93.1%	10.3	81.6%	11.3	88.2%	**11.8**	92.8%	11.4	97.0%	12.6
Unfriendly	0%	88.9%	9.7	80.2%	3.7	85.3%	**9.9**	89.2%	9.8	94.2%	11.5
	15%	82.5%	7.9	74.8%	-0.3	78.8%	6.4	83.4%	**8.2**	92.4%	9.3
	30%	79.7%	6.7	69.3%	-2.1	73.7%	5.0	81.1%	**7.0**	89.8%	8.6

Table 4.6 compares the policy models on the basis of reward and success rates after 4000 training dialogues. Each row represents a different task with the user-type provided by the user simulator and the SER set by the error model. The first three rows compare the performance of policy learning models with data generated by the standard user simulator under 0% SER introduced by the error model. As the standard user simulator carries fruitful interactions with the policy by providing all the required pieces of information, the performance in terms of success rate is comparatively higher than the unfriendly user. The impact of SER is also evident in the performance of the models. When the percentage (%) of SER increases, the average success rates and average rewards get reduced.

The GP-SARSA shows the highest success rates in all six environmental settings. This is due to its superiority in modelling the uncertainty with efficient exploration/exploitation approximation kernel function under small size application domain. In addition, it is a non-parametric method and performs optimisation in value space rather than policy space. It makes it less effective in high-dimensional or continuous action spaces because when the space is large, the usage of memory and computation consumption grows rapidly. DQN achieves the lowest performance due to high-variance in the samples as well as often converges to local-optima. This is why its performance is highly unstable.

In deep learning-based RL approaches, actor-critic based models show more stable performance and are comparable to others in all the environment settings. In addition, the incorporation of experience replay clearly enhances the actor-critic performance design A2CER policy learning. Overall, the results validate the benefit of data-driven deep learning-based policy learning for dialogue, where the system can effectively be pre-trained using the data generated by a user-simulator under several challenging environmental modes and then can further be refined via real-time user interactions.

The experiment shows that A2CER's performance is comparable to GP-SARSA in terms of sample efficiency, speed of convergence, success rate, rewards and the number of turns. However, the success rate of A2CER remains 1-3 percentage points lower than the GP-SARSA; A2CER needs fewer dialogue samples to train and eventually achieves higher rewards compared to GP-SARSA. This is because the A2CER algorithm optimises the reward function rather than the success rate, which leads to a slightly lower success rate. Moreover, A2CER performs well compared to other neural network-based RL methods in terms of success rate, sample efficiency, speed of convergence and rewards.

Lastly, It is worth noting the performance of Rule-Based policy. In almost all the tasks, it outperforms all the RL-based policies in our Restaurant domain. It shows that the data-driven RL-based models still suffer from the issues of large state spaces learning. However, the state space abstraction [285, 286] can be utilised to mitigate the issue; the problem is open to future research in this area.

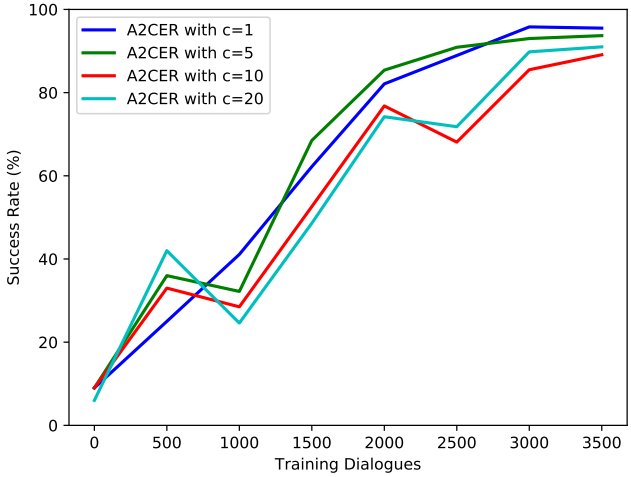


Figure 4.7 Success rate of A2CER with varying hyperparameter c.

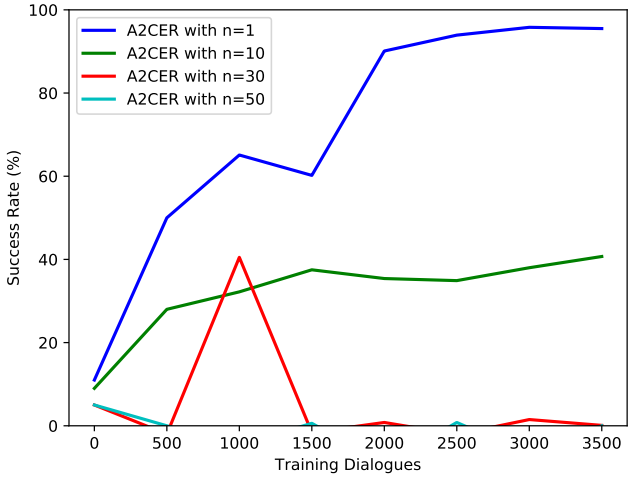


Figure 4.8 Success rate of A2CER with varying hyperparameter n.

4.6.2 Hyperparameter Tuning

This section investigates the effect of hyperparameters c and n on the performance of the proposed A2CER algorithm. The c value sets the upper bound for IS weight. The estimated weight higher than c is truncated. It helps in setting up the accurate bias correction term. A very high value of c ignores the truncation effect, while a too low value leads to the introduction of a less accurate bias correction term. From Figure 4.7, we observe that c=5 achieves the highest success rate and overall good performance. In addition, it is also evident that for a wide range of values c=1 to c=20, there is no significant change in the final performance. It shows that this hyperparameter does not have much influence on the algorithm's performance. Figure 4.8, on the other hand, explore the A2CER's performance on the number of training steps per episode n. It is observed that the training diverges substantially when n is too high due to rapid change in the policy value. When n=1, the algorithm converges quickly, and performance is also good, while n=10, the performance is consistently poor. For n=30 and n=50, the algorithm diverges completely.

4.6.3 Learning from Demonstration Data

Training a dialogue policy from scratch always draws a poor user experience in the initial stage until a sufficient amount of interactions have been performed, achieving the acceptable behaviour for a system no matter what model or learning algorithm we use. As mentioned in Section 4.5.4, a demonstration data as an off-line corpus can help in mitigating this problem. To investigate this, we utilise the HDRS corpus [1] containing 1400 real user dialogues in the Allahabad Restaurant domain (see Appendix A for the brief description of the domain). The corpus was divided into 4:1:1 ratio of training, testing and development sets.

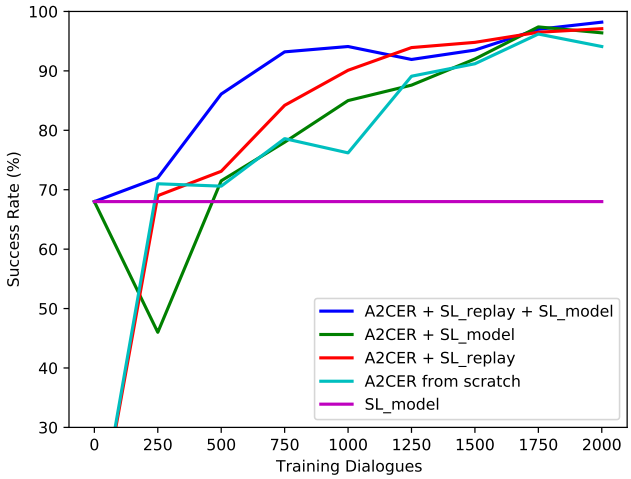


Figure 4.9 Learning curve of A2CER policy with demonstration data.

Figure 4.9 shows the impact of demonstration data in policy learning using A2CER in a noise-free condition (0% SER). We use the demonstration data in two ways aiming to achieve improved performance: 1) use it to pre-train the model (SL_model), 2) use it as initial experience replay (SL_replay), or 3) both. Success rates over four combinations of A2CER policy learning and a SL_model based policy are drawn for the investigation. The A2CER model followed by the supervised training (A2CER+SL_model) shows the improvement only after 600 interactions on-line with the users after sufficient interactions. This is because the pre-trained parameters obtained from the optimised SL are quite distinct from the optimal parameters of A2CER. Using the demonstration data as a replay buffer (A2CER+SL_replay) shows better performance otherwise when the model (A2CER) is trained from scratch. Additionally, the combined role of SL pre-training and SL replay (A2CER+SL_model+SL_replay) achieves the best result by encompassing the two-fold benefit of demonstration data. It is evident that the use of demonstration data provides an initial boost in the performance of the A2CER algorithm compared to learning it from scratch.

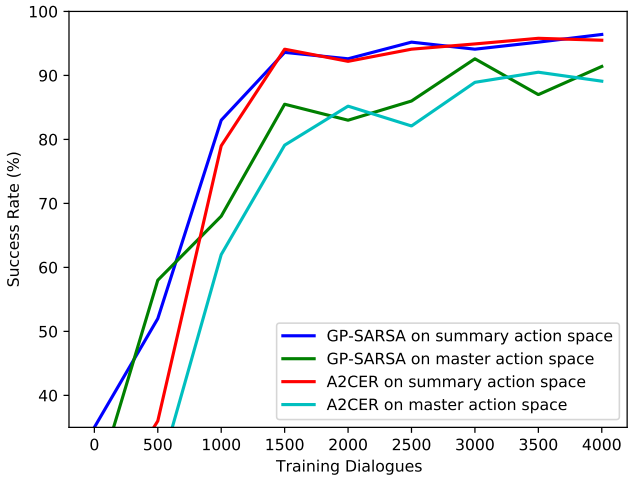


Figure 4.10 Success rate of A2CER and GP-SARSA on summary and master action spaces.

4.6.4 Comparison on Master and Summary Action Space

In our experiments, A2CER delivers comparable performance among NN-based RL methods but performs almost equally if not slightly worse than the GP-SARSA. The previous experiments were done on the summary action space containing only 15 actions. GP-SARSA suffers from high prohibitive computational costs in a magnitude order of more actions as it needs to invert the Gram matrix [124] before predicting each response. On the other hand, the A2CER might be useful in such scenarios because it does not have the prohibitive computation cost, and is supposed to train efficiently in a very less time than GP-SARSA.

To prove the hypothesis, we experiment with A2CER and GP-SARSA and compare their performance both on summary and master action spaces in Figure 4.10. Both A2CER and GP-SARSA suffer from slow convergence on master action space. It was anticipated because the system has to choose one from a set of large actions in the master action space (205 in comparison with 15 summary actions). Additionally, the random initialisation of a policy will be less reasonable on the master action space than on the summary space; the latter has the advantage of hard-coded heuristic-based mapping from summary to master action.

It is evident that both A2CER and GP-SARSA are able to perform well with the large action spaces quite efficiently. Due to better sample efficiency, GP-SARSA achieves improved performance than A2CER on the challenging master action space. Despite this, it takes a huge amount of time to train due to the requirement of vast computational resources to run. When trained on master action space, A2CER took 4.5 hours to learn, while GP-SARSA ran for 5.3 days to complete the training. With a slightly lower success rate, A2CER is more reasonable than the GP-SARSA in terms of the computational cost and suitable to be utilised in real-time interactions on a reasonably sized domain.

Table 4.7 Human Evaluation.

	GP-SARSA	A2CER
Success rate	91.8%	91.2%
Avg Reward per dialogue	11.34(±7.67)	12.43(±8.35)
Avg Turns per dialogue	5.46(±3.29)	4.38(±2.71)

4.6.5 Human Evaluation

In the previous sections, the models are analysed in a simulated environment. To test the generalised performance of the proposed methods, we set up and evaluate the trained dialogue policies in real-time interactions with users as suggested in [287]. We recruited some students and used our WOZ setup [1] who can voluntarily interact with our dialogue system and rate it. We used two dialogue policies, i.e. GP-SARSA and A2CER. They were trained on summary action space with 15% SER to be capable of handling sufficient ASR errors. The learnt policies are then incorporated into the SDS pipeline with commercial ASR and TTS systems. The users were asked to interact with the system aiming to find restaurants based on the particular features of the given task. In the experiment, subjects were uniformly allocated to each analysed systems (dialogue policy method). Upon completing the dialogue, the users were asked to judge the conversation whether it was successful or not. Table 4.7 presents the success rate, mean of rewards and number of turns obtained from both systems. In terms of success rate, both dialogue policies perform well. However, A2CER achieves considerably higher rewards under smaller average turns during the interactions.

4.7 Summary

The chapter presents the problem of building a dialogue policy in the Hindi domain under a task-oriented environment, which is solved by adapting two categories of RL methods, i.e. value-based and policy-based. The contributions lie in the following points:

- The chapter demonstrates the effectiveness of RL in learning a dialogue policy and gives a brief discussion of modelling the dialogue as a POMDP. It also discusses the limitations of value-based methods.
- The recent introduction of deep learning in RL methods has also been investigated, which comes under the policy-based category especially gradient-based methods.
- In the experimental setup, we describe the components, i.e. user simulator, dialogue evaluation and reward estimation and models undertaken like value-based methods, i.e. GP-SARSA, DQN, policy-based DRL methods, i.e. A2C, followed by our proposed A2CER.

- We prove that our version of A2C with experience replay achieves better performance than the current state-of-the-art NN-based policy learning methods. It is found to be more sample-efficient and broadly competitive with GP-SARSA in terms of success rates and the average of collected rewards.

However, A2CER still lags behind the GP-SARSA, as the currently used application domain is relatively small. A2CER will surely beat when experimented on very large-domain spaces. Additionally, we have also shown how the demonstration data can be helpful in mitigating the model's early-stage performance issues. A2CER is found to be effective with this setting.

Whether training is being performed on a summary or master action space, both are the set of static action only. The major limitation under this framework is that the entire policy must be relearned from scratch when a new action space or new domain-ontology schema is introduced. It would be a serious limitation in maintaining a real-life dialogue system, as it requires a regular change in the action space and database schema. Hence, the training algorithms need to be devised that are able to keep their preexistent knowledge and capable of adapting new changes in the framework, which is an important and vital area to investigate in the future. There are many possible ways to explore it within the framework of DRL based dialogue modelling [288].

Chapter 5

Hindi Dialogue Generation

5.1 Introduction

The task of the Natural Language Dialogue Generation (NLDG) module in a task-oriented SDS is to produce a natural, meaningful sentence on a specified *Dialogue-Act* (DA) [136, 29]. A dialogue act has the details of action to be performed, i.e. *inform* or *request* accompanied with one or more slot-value pairs, i.e. `inform(name="महाराजा तंदूरी रेस्टोरेंट", near="एम जी मार्ग", kidsallowed="yes", food="नार्थ इंडियन")` as shown:

D.Act:	`inform(name="महाराजा तंदूरी रेस्टोरेंट", near="एम जी मार्ग", kidsallowed="yes", food="नार्थ इंडियन")`
G.Utterance:	महाराजा तंदूरी रेस्टोरेंट एम जी मार्ग में है वहाँ नार्थ इंडियन खाना मिलता है और बच्चों को प्रवेश की अनुमति है।

D.Act: System DA to be converted into the natural sentence.

G.Utterance: Generated system utterance.

A natural language dialogue generator must be capable of producing semantically and syntactically correct utterances. In order to draw a natural conversation, NLDG systems should express all the information presented in an input DA. In a general architecture, the NLDG task is carried out in two phases: 1. *sentence planning* and 2. *surface realisation*. The sentence planning phase handles the generation of intermediate structures, i.e. Bayesian network, dependency trees or templates from the semantic input (DA) [136, 29]. Later, this intermediate form will be realised as the final natural language response in the surface realisation phase.

Initially, most NLDG systems were based on *rule-based* approaches [131, 28] or a hybrid of handcrafted and statistical methods [132, 134, 133]. For example, the first statistical NLDG model, *HALogen*[1], was implemented by Langkild et al., which performs reranking on handcrafted candidates using an n-gram Language Model (LM)

[1] HALogen is a successor to Nitrogen [132].

[134]. In 2000, a class-based n-gram language model generator, a type of *word-based generator*, was proposed to generate sentences stochastically for a task-oriented dialogue system [27]. However, inherently it has a very high computation cost, and it is indefinite about covering all possible semantics in the outputs. Hence later, the word-based generators were replaced by *phrase-based generators* which had not only reduced the computation cost but also generated linguistically varied utterances [136, 29]. However, the phrase-based generators are restricted to semantically-aligned corpora which are tedious and expensive to collect.

More recently, researchers have used methods that do not require aligned data and perform *end-to-end* training to get sentence planning and surface realisation done in one go [137]. For achieving the naturalness, variation and scalability on unaligned corpora, they incorporated the deep-learning models. The successful approaches use the RNN-based models to train the *encoder-decoder* on a corpus of paired DAs and corresponding utterances [32, 33]. Wen et al. proposed various Recurrent Neural Network Language Generation (RNNLG) models, i.e. Attention-Based Encoder-Decoder (ENC-DEC), Heuristically-gated LSTM (H-LSTM) and Semantically Controlled LSTM (SC-LSTM) which are also shown to be effective for the NLDG module in task-oriented dialogue systems [12, 138]. Although the deep-learning methods are supposed to learn a high level of semantics, but they require a large amount of data for even a small task-oriented system.

Furthermore, in the rule-based and statistical models, e.g. n-gram and K-Nearest Neighbors (KNN), the NLDG module in an SDS considers only the provided DA as input and can not adapt to the user's way of speaking. People have tried just not only to avoid the repetition but also to add variations into the generated responses, typically, either by alternating over a *pool of preset responses* [289], selecting randomly over *k-best generated samples* or using *overgeneration* [12]. The concept of *entrainment* has also been introduced recently into NLDG in SDS to enhance the perceived naturalness of the response, but they are primarily rule-based [290]. However, we have observed that none of the approaches has been investigated on a *Hindi-corpora*.

In this chapter, we have explored several RNNLG-based models: (a) H-LSTM, (b) SC-LSTM, (c) Modified Semantically Controlled LSTM (MSC-LSTM), (d) ENC-DEC, (e) SC-RNN, (f) H-RNN and (g) Vanilla-LSTM (V-LSTM) and compared them with the benchmark models, i.e. Hand-Crafted (HDC), KNN model and n-gram model. All the models are experimented on our own Hindi dialogue dataset, collected on the restaurant domain. The modified RNNLG-models with the proposed dataset are released at the following URL:

https://github.com/skmalviya/RNNLG-Hindi

The chapter is organised into six sections: current Section 5.1 presents introduction of the chapter discussing the NLDG task and related work. Section 5.2 and 5.3, described the baseline NLDG models and RNNLG framework based models respectively. The experimental studies with dataset description and results & analysis are presented in Section 5.4 and 5.5, respectively, followed by Section 5.6, which mentions the conclusion & possible future extensions.

5.2 Baseline NLDG Models

This section describes baseline-NLDG models, e.g. n-gram and KNN. In the next section, V-LSTM, H-RNN, H-LSTM, ENC-DEC, SC-RNN, SC-LSTM and MSC-LSTM are discussed, which are variants of recurrent-neural models.

5.2.1 Class-Based *n*-gram Model

It first divides the data over each existing *combinations*[2] of action and slot-value pairs. Then it estimates an n-gram model separately for each class. This work chooses n-gram = 5 as the maximum n-gram window size. It generates the utterances based on the probability distribution of tokens in delexicalised utterances [27]. The model decodes (generates) utterances with beam-search decoding.

Suppose an utterance is to be generated on a DA-class d. The model generates a set of candidate-output sentences through the language model trained for the DA-class d. Based on the probability of an utterance $P(X)$, the most likely output candidate is selected through Equation 5.1:

$$X = \arg\max_X P(X|d) = \arg\max_X P(X) \tag{5.1}$$

where, X denotes an output sentence as $X = (x_1, x_2, ..., x_T)$ with a sequence of words $x_1, x_2, ..., x_T$ and d is the desired class for which the sentence is to be generated. The class term d is common to all the generations, so it can be discarded. Thus, the $P(X)$ is estimated by the chain-rule as below:

$$P(X) = P(x_1, x_2, ..., x_T) = \prod_{t=1}^{T} P(x_t|x_1, x_2, ...x_{t-1}) \tag{5.2}$$

where, $P(x_t|x_1, x_2, ...x_{t-1})$ is calculated using the n-gram exist in the training corpus:

$$P(x_1, x_2, ..., x_T) = \frac{\text{count}(x_{t-n}, ..., x_{t-1}, x_t)}{\text{count}(x_{t-n}, ..., x_{t-1})} \tag{5.3}$$

where, count$(x_{t-n}, ..., x_{t-1}, x_t)$ denotes the number of times a sequence of terms (n-gram) occurs in the dataset.

5.2.2 KNN Model

This model is based on the K-Nearest Neighbors (KNN) classifier. It generates an utterance with the help of training and validation data from the corpus. First, it constructs a 1-hot DA vector and finds its ($da1$) similarity da_sim (as shown in Equation 5.4) with all DA vectors ($da2$) of training and validation data to generate top-n

[2] It is accomplished by assigning each DA to a group of similar physical appearance.

delexicalised utterances based on the similarity score. Later, it reranks the delexicalised utterances on the basis of Total Error (Slot Error + Binary-Value Error) and then lexicalises it for the surface-realisation.

$$\text{da\_sim} = \frac{\text{len}\big(A(da1) \cap A(da2)\big) + \text{len}\big(SV(da1) \cap SV(da2)\big)}{\sqrt{\text{len}\big(A(da1) + A(da2)\big)} * \sqrt{\text{len}\big(SV(da1) + SV(da2)\big)}} \tag{5.4}$$

where, len() shows the length of a set, $A()$, $SV()$ represents a set of actions and slot-value pairs respectively of a DA. it is simple and easy to implement but computationally inefficient as it requires a huge number of comparisons for each generation.

5.3 RNNLG Models

This section describes various *RNNLG framework* based NLDG models [138]. The use of RNN in language generation got inspired after it was successfully applied in sequential data prediction through the RNN based Language Modelling (RNNLM) [68, 291]. It was proved that RNNLM learns the output distribution on the previous word as input and generates syntactically correct sequences. However, the sequential output does not ensure *semantic coherency*. Hence, to generate appropriate sequences, RNN based models are required to be conditioned on semantic details as well.

Basically, in RNNLG, a generator takes DA as an input, comprised of *DA-type*, i.e., inform, request, affirm, etc., and a set of *slot-value* pairs and generates an appropriate utterance in return. As shown in Figure 5.1, the DA '`inform(name="राजवाड़ा", area="कर्नल गंज")`' is given as input and the generated output is 'राजवाड़ा कर्नल गंज में है।'. The framework comprises of two parts: first a DA-cell which deals with *content-planning* (semantics) that updates the DA-vector[3] at each time-step either stochastically or heuristically and second an sequence-to-sequence (s2s) cell that deals with *surface-realisation* (utterance-generation) and updates the hidden-vector conditioned on both delexicalised utterance as well as current DA-vector as in Equation 5.5.

Before training an s2s-cell, the data is first *delexicalised* by replacing the values of slots with the specified slot-tokens[4], to make efficient use of training data. The network output is a sequence of tokens that can be lexicalised for the appropriate surface realisation.

Typically, a model in RNNLG framework takes *Word2Vec* embedding $\mathbf{w}_t$ of a token w_t as an input at each time step t in conditioned with the previous hidden state $\mathbf{h}_{t-1}$ and update it as $\mathbf{h}_t$ for the next iteration to predict the next token w_{t+1} cf., Equation. 5.6 and 5.7. Furthermore, DA's encoding $\mathbf{d}_t$ is also encapsulated in the RNN in order to embed DA's effect in the generation. Hence, the recurrent function f that updates the hidden state is represented as:

[3]DA-vector is a 1-hot encoded vector of action-type and slot-value-type where values are corresponding to occurrences of a given slot e.g. sv.name._1, sv.name._2.

[4]Here, token is used to represent both word and slot-token e.g. SLOT_NAME, SLOT_AREA etc. in a delexicalised sentence.

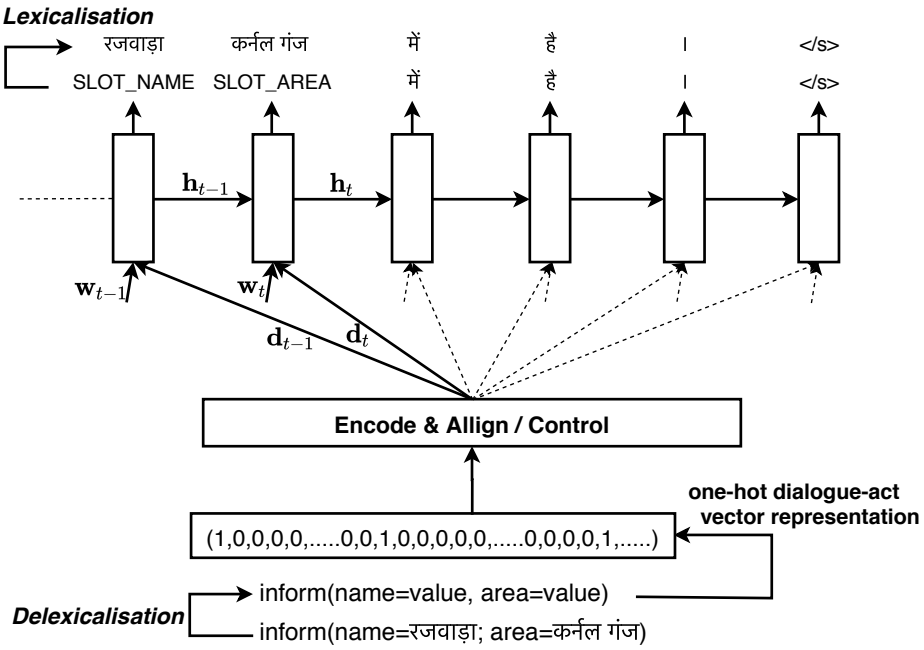


Figure 5.1 Basic RNNLG Framework

$$\mathbf{h}_t = f(\mathbf{w}_t, \mathbf{h}_{t-1}, \mathbf{d}_t) \tag{5.5}$$

This updated hidden state is then transformed to an output probability distribution, from which the next token in the sequence is sampled by function f with softmax-encoding as in Equation 5.6. After training the model at each time step t, the output is generated through beam-search decoding as in Equation 5.7:

$$P(w_{t+1}|w_t, w_{t-1}, ..., w_0, \mathbf{d}_t) = softmax(\mathbf{W}_{ho}\mathbf{h}_t) \tag{5.6}$$

$$w_{t+1} \sim P(w_{t+1}|w_t, w_{t-1}, ..., w_0, \mathbf{d}_t). \tag{5.7}$$

The different approaches for generating hidden-vector in *RNN/LSTM cell* and generating updated DA-vector in *DA-cell* are explained in the following sub-sections. The simplest model of this framework is Vanilla-LSTM which takes aligned 1-hot DA-vector. So it does not require additional DA-cell.

5.3.1 Vanilla-LSTM

A type of gated RNN architecture, Vanilla-LSTM (V-LSTM) stores more relevant information with the help of a memory-cell $\mathbf{c}_t$ and a set of pointwise multiplicative/additive gates to regulate how the memory-cell is going to be updated, stored and used subsequently [17, 292]. In each iteration, input token $\mathbf{w}_t$ and previous hidden state $\mathbf{h}_{t-1}$ and previous cell-state $\mathbf{c}_{t-1}$ are given to LSTM-cell and updates its internal states as follows:

$$\mathbf{i}_t = \sigma(\mathbf{W}_{wi}\mathbf{w}_t + \mathbf{W}_{hi}\mathbf{h}_{t-1} + \mathbf{W}_{di}\mathbf{d}_t) \tag{5.8}$$

$$\mathbf{f}_t = \sigma(\mathbf{W}_{wf}\mathbf{w}_t + \mathbf{W}_{hf}\mathbf{h}_{t-1} + \mathbf{W}_{df}\mathbf{d}_t) \tag{5.9}$$

$$\mathbf{o}_t = \sigma(\mathbf{W}_{wo}\mathbf{w}_t + \mathbf{W}_{ho}\mathbf{h}_{t-1} + \mathbf{W}_{do}\mathbf{d}_t) \tag{5.10}$$

$$\hat{\mathbf{c}}_t = \sigma(\mathbf{W}_{wc}\mathbf{w}_t + \mathbf{W}_{hc}\mathbf{h}_{t-1} + \mathbf{W}_{dc}\mathbf{d}_t) \tag{5.11}$$

$$\mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \hat{\mathbf{c}}_t \tag{5.12}$$

$$\mathbf{h}_t = \mathbf{o}_t \odot \tanh(\mathbf{c}_t) \tag{5.13}$$

here, σ and $\tanh$ are the activation functions. Using the $\mathbf{W}_{*,*}$ model parameters, $\mathbf{i}_t$, $\mathbf{f}_t$ and $\mathbf{o}_t$ estimate the typical input, forget and output gates values. $\odot$ performs the element-wise multiplication. As per the V-LSTM architecture, current cell value $\hat{\mathbf{c}}_t$ after getting modified by the gates' values, will generate the true cell-value $\mathbf{c}_t$ at time t.

This is the Vanilla-LSTM implementation. The remainder of the section explains other RNNLG framework models capable of incorporating context information that ensures the meaning of the surface-output be consistent with the input DA. We have explored various ways to implement the recurrent function f such as H-LSTM, SC-LSTM, MSC-LSTM (proposed) and ENC-DEC [138]. All the models follow a typical two-cell architecture as mentioned earlier, first a *DA-cell* to model the semantic-input $\mathbf{d}_t$ and second an *LSTM-cell* for updating the hidden-vector $\mathbf{h}_t$.

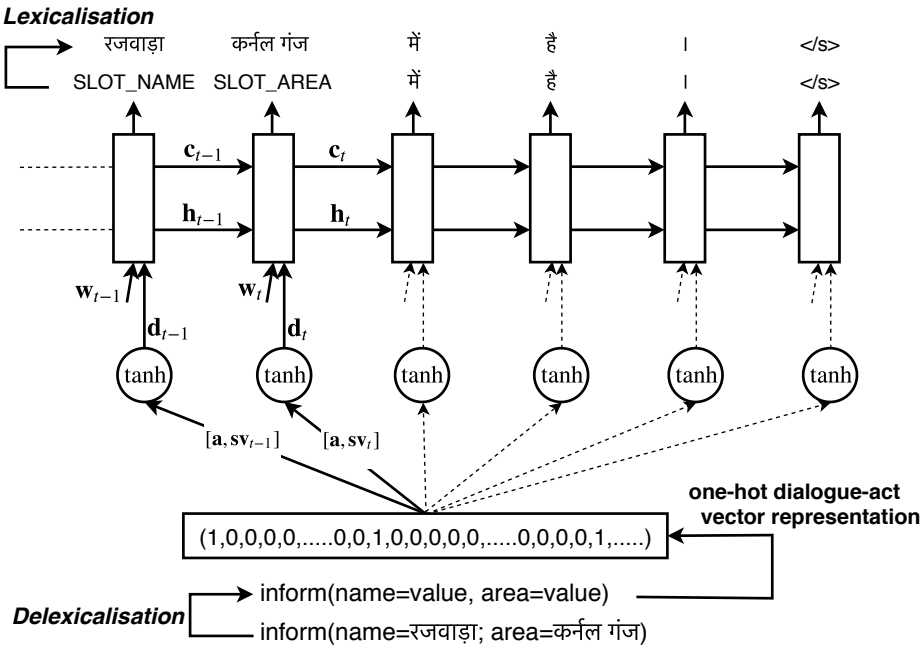


Figure 5.2 The architecture of H-LSTM Model.

5.3.2 Heuristically-Gated Model (H-RNN, H-LSTM)

The *heuristic* approach for DA-cell was proposed to incorporate the current DA-information $\mathbf{d}_t$ in the form of a 1-hot encoding vector of DA-type and slot-value pairs to generate meaningful utterances [12]. The architecture of the H-LSTM model is shown in Fig. 5.2. In order to avoid the undesirable repetitions in the generation, $\mathbf{d}_t$ is reapplied together with the $\mathbf{w}_t$ through a *heuristic reading-gate* $\mathbf{r}_t$ in DA-cell by Equation 5.14 and 5.15. The reading-gate's task is to update $\mathbf{d}_{t-1}$ such that if any slote-token, e.g. SLOT_NAME, SLOT_AREA, DONT_CARE etc. appeared in the last step, then the index of corresponding slot-value in $\mathbf{d}_{t-1}$ was set to zero with the help of Equation 5.14:

$$\mathbf{d}_t = \mathbf{r}_t \odot \mathbf{d}_{t-1} \tag{5.14}$$

$$\mathbf{d}_t = \tanh(\mathbf{W}_{rd}\mathbf{d}_t) \tag{5.15}$$

Due to the vanishing gradient problem in long sentences, an improved version of LSTM is used as an s2s model and is called H-LSTM (Heuristically-gated LSTM) as it takes *heuristically modified* $\mathbf{d}_t$ as input in each time-step:

$$\mathbf{i}_t = \sigma(\mathbf{W}_{wi}\mathbf{w}_t + \mathbf{W}_{hi}\mathbf{h}_{t-1} + \mathbf{W}_{di}\mathbf{d}_t) \tag{5.16}$$

$$\mathbf{f}_t = \sigma(\mathbf{W}_{wf}\mathbf{w}_t + \mathbf{W}_{hf}\mathbf{h}_{t-1} + \mathbf{W}_{df}\mathbf{d}_t) \tag{5.17}$$

$$\mathbf{o}_t = \sigma(\mathbf{W}_{wo}\mathbf{w}_t + \mathbf{W}_{ho}\mathbf{h}_{t-1} + \mathbf{W}_{do}\mathbf{d}_t) \tag{5.18}$$

$$\hat{\mathbf{c}}_t = \sigma(\mathbf{W}_{wc}\mathbf{w}_t + \mathbf{W}_{hc}\mathbf{h}_{t-1} + \mathbf{W}_{dc}\mathbf{d}_t) \tag{5.19}$$

$$\mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \hat{\mathbf{c}}_t \tag{5.20}$$

$$\mathbf{h}_t = \mathbf{o}_t \odot \tanh(\mathbf{c}_t) \tag{5.21}$$

where, $\mathbf{W}_{*,*}$ represents training weights and $\mathbf{i}_t$, $\mathbf{f}_t$, $\mathbf{o}_t$ are the input, forget and output gates of the LSTM-cell and $\hat{\mathbf{c}}_t$, $\mathbf{c}_t$ denotes step-wise local and global vector of the memory-cell. $\odot$ performs the element-wise multiplication.

Both RNN and LSTM models with heuristically-gated architecture are used for the surface-realisation. In H-RNN, the hidden-vector $\mathbf{h}_t$ is updated by:

$$\mathbf{h}_t = \sigma(\mathbf{W}_{wh}\mathbf{w}_t + \mathbf{W}_{hh}\mathbf{h}_{t-1} + \mathbf{W}_{dh}\mathbf{d}_{t-1}) \tag{5.22}$$

H-RNN's architecture is depicted in Figure 5.1.

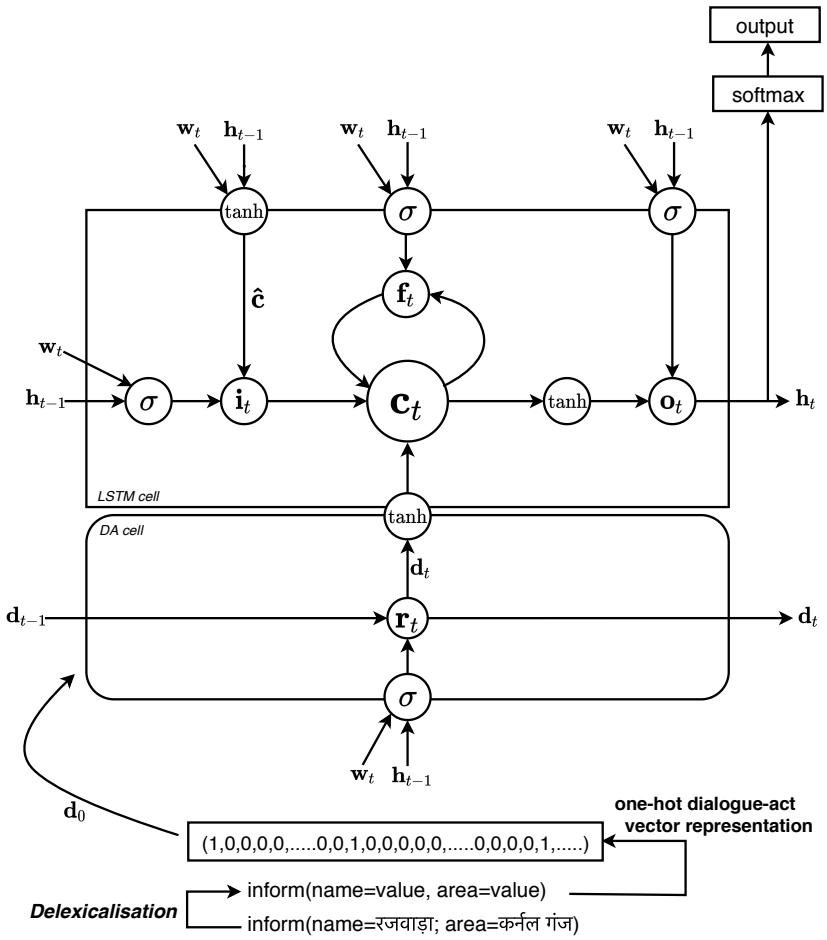


Figure 5.3 The architecture of SC-LSTM Model

5.3.3 Semantically-Controlled Models (SC-LSTM, SC-RNN)

As we see, H-LSTM perfectly models the delexicalised data and incorporates DA details accurately up to a level in the generation. But this simple *content-planning* ability does not make it capable of handling the *binary-slots*, e.g. *kidsallowed=yes* and the slots assigned with 'DONT_CARE' value, which can not be delexicalised. It is evident that the direct one-to-one matching of slot-value pairs and the corresponding surface-form realisation is not possible in H-LSTM. To address this issue, a mechanism for reading-gate $\mathbf{r}_t$ in the DA-cell is proposed by Wen et al. [12] (as shown in Fig. 5.3), which remembers the associated phrases corresponding to slot-value pairs stochastically.

$$\mathbf{r}_t = \sigma(\mathbf{W}_{wr}\mathbf{w}_t + \mathbf{W}_{hr}\mathbf{h}_{t-1} + \mathbf{W}_{dr}\mathbf{d}_{t-1}) \tag{5.23}$$

where, $\mathbf{W}_{wr}$, $\mathbf{W}_{hr}$ and $\mathbf{W}_{dr}$ are again the weight matrix to learn the sequential pattern of both key-phrases and the associated slot-values. Thus, the updated reading gate as in Equation 5.23, if replaced in Equation 5.14, would make the model more resilient to learn delexicalised phrases.

For surface realisation, both RNN and LSTM cells are used separately as language generation model of SC-RNN and SC-LSTM. In SC-RNN, the final output depends only on the hidden vector as presented in Figure 5.4:

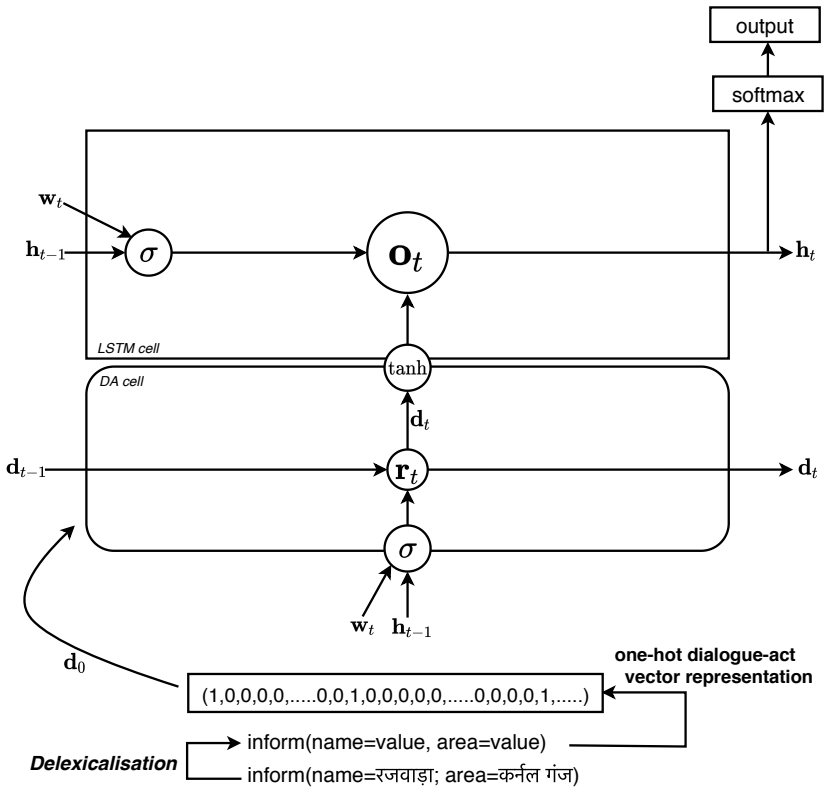


Figure 5.4 The architecture of SC-RNN Model.

$$\mathbf{h}_t = \mathbf{o}_t = \sigma(\mathbf{W}_{wo}\mathbf{w}_t + \mathbf{W}_{ho}\mathbf{h}_{t-1}) + \tanh(\mathbf{W}_{do}\mathbf{d}_t) \tag{5.24}$$

Build on a typical LSTM architecture, SC-LSTM also has a memory-cell (see Figure 5.3). The model updates the hidden layer as follows:

$$\mathbf{i}_t = \sigma(\mathbf{W}_{wi}\mathbf{w}_t + \mathbf{W}_{hi}\mathbf{h}_{t-1}) \tag{5.25}$$

$$\mathbf{f}_t = \sigma(\mathbf{W}_{wf}\mathbf{w}_t + \mathbf{W}_{hf}\mathbf{h}_{t-1}) \tag{5.26}$$

$$\mathbf{o}_t = \sigma(\mathbf{W}_{wo}\mathbf{w}_t + \mathbf{W}_{ho}\mathbf{h}_{t-1}) \tag{5.27}$$

$$\hat{\mathbf{c}}_t = \sigma(\mathbf{W}_{wc}\mathbf{w}_t + \mathbf{W}_{hc}\mathbf{h}_{t-1}) \tag{5.28}$$

$$\mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \hat{\mathbf{c}} + \tanh(\mathbf{W}_{dh}\mathbf{d}_t) \tag{5.29}$$

$$\mathbf{h}_t = \mathbf{o}_t \odot \tanh(\mathbf{c}_t) \tag{5.30}$$

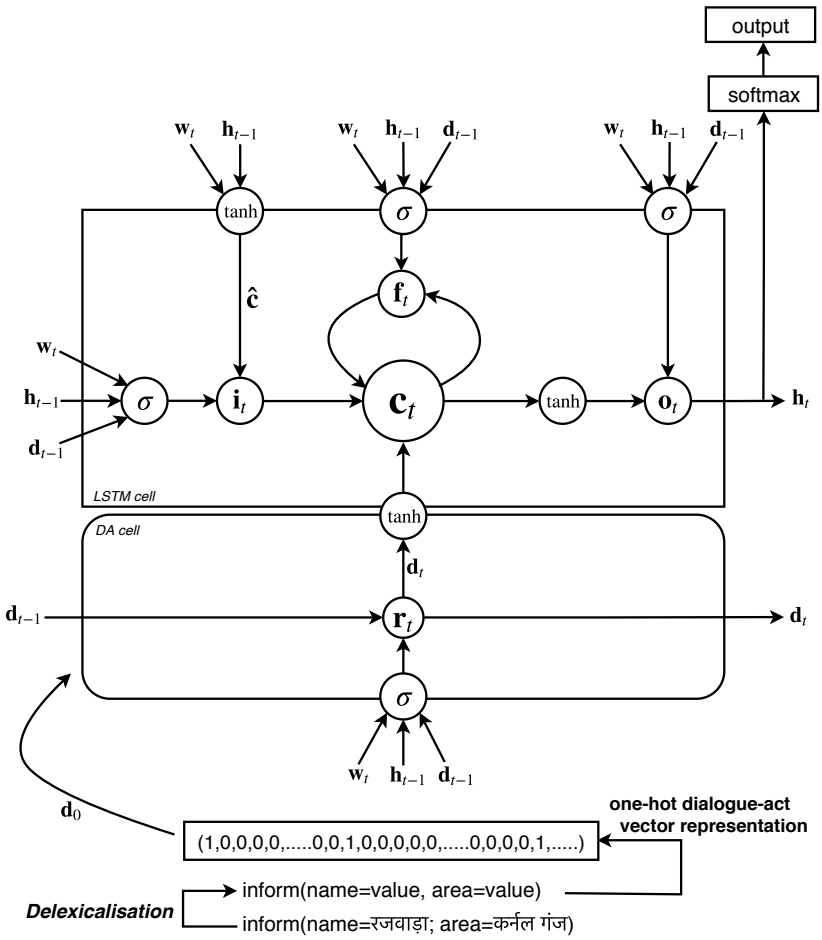


Figure 5.5 The architecture of MSC-LSTM Model.

5.3.4 Modified-Semantically-Controlled Model (MSC-LSTM)

We proposed a modified version of SC-LSTM (MSC-LSTM) that includes the influence $\mathbf{d}_t$ not only in estimating the memory-cell value but also in readjusting the weights of input, forget and output gates. MSC-LSTM shows better performance than SC-LSTM and H-LSTM, as discussed in the result section 5.5. DA-cell is the same as in SC-LSTM. In LSTM-cell now, the current DA-vector (without DA-act type) is conditioned on each input, forget and output gate beside the memory-cell as in Figure 5.5. So, the modification is only in the estimation of $\mathbf{i}_t$, $\mathbf{f}_t$ and $\mathbf{o}_t$:

$$\mathbf{i}_t = \sigma(\mathbf{W}_{wi}\mathbf{w}_t + \mathbf{W}_{hi}\mathbf{h}_{t-1} + \mathbf{W}_{di}\mathbf{d}_{t-1}) \tag{5.31}$$

$$\mathbf{f}_t = \sigma(\mathbf{W}_{wf}\mathbf{w}_t + \mathbf{W}_{hf}\mathbf{h}_{t-1} + \mathbf{W}_{df}\mathbf{d}_{t-1}) \tag{5.32}$$

$$\mathbf{o}_t = \sigma(\mathbf{W}_{wo}\mathbf{w}_t + \mathbf{W}_{ho}\mathbf{h}_{t-1} + \mathbf{W}_{do}\mathbf{d}_{t-1}) \tag{5.33}$$

$$\hat{\mathbf{c}}_t = \tanh(\mathbf{W}_{wc}\mathbf{w}_t + \mathbf{W}_{hc}\mathbf{h}_{t-1}) \tag{5.34}$$

$$\mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \hat{\mathbf{c}} + \tanh(\mathbf{W}_{dh}\mathbf{d}_t) \tag{5.35}$$

$$\mathbf{h}_t = \mathbf{o}_t \odot \tanh(\mathbf{c}_t) \tag{5.36}$$

5.3.5 Attention-Based Encoder-Decoder (ENC-DEC)

This model is based on the architecture proposed earlier for Neural Machine Translation [31] which takes attention from all the slot-value during the encoding such that the sum of *attention-distribution* is equal to one, as shown in Figure 5.6. The $\mathbf{i}^{th}$ slot-value pair $\mathbf{z}_i$ is represented by the sum of distributed vectors of slot-embeddings $\mathbf{s}_i$ and value-embeddings $\mathbf{v}_i$:

$$\mathbf{z}_i = \mathbf{s}_i + \mathbf{v}_i \tag{5.37}$$

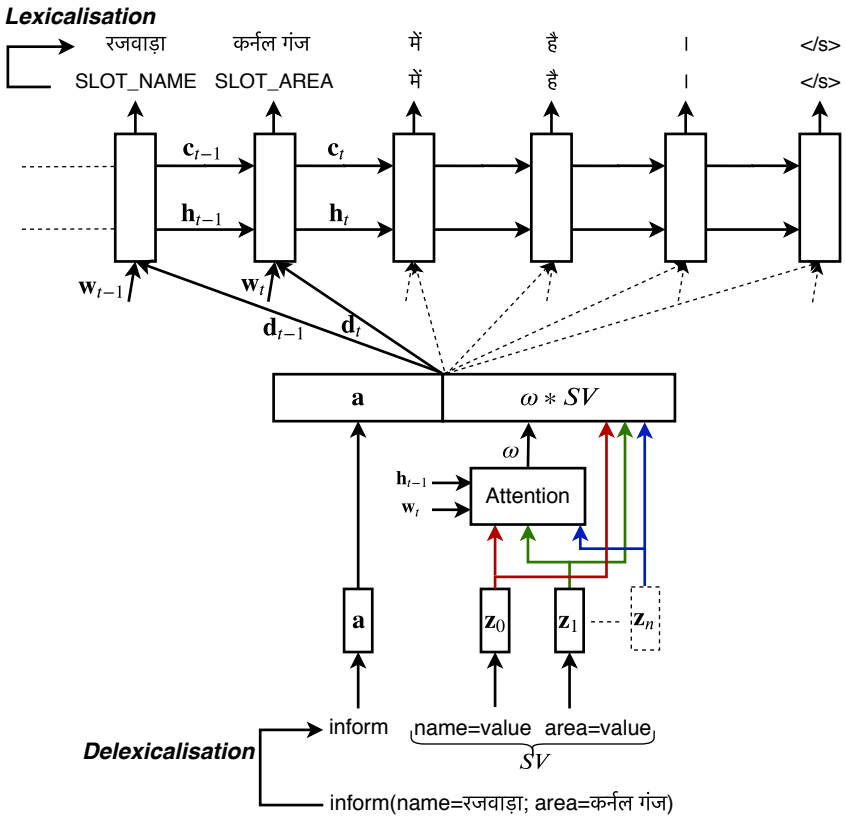


Figure 5.6 The architecture of Attention-Based ENC-DEC Model.

where, the size of vectors $\mathbf{s}_i$ and $\mathbf{v}_i$ is set equal to hidden-vector size. For adding semantics to $\mathbf{d}_t$, the slots and values are separately represented as a parameterised vector [33]. At each time-step t, the output of the DA-cell $\mathbf{d}_t$ is estimated by $\mathbf{z}_i$, weighted to its attentions:

$$\mathbf{d_t} = \mathbf{a} \oplus \sum_i \omega_{t,i} \mathbf{z_i} \tag{5.38}$$

where $\mathbf{a}$ is the vector representation of DA-type and $\oplus$ is the sign of concatenation, and $\omega_{t,i}$ is the attention-weight of i^{th} slot-value pair. At each iteration, the attention of all the slot-value pairs are calculated as $\beta_{t,i}$ and normalised between 0 and 1 as $\omega_{t,i}$ by Equation 5.39 and 5.40 respectively:

$$\beta_{t,i} = \mathbf{q}^T \tanh(\mathbf{W}_{hm}\mathbf{h}_{t-1} + \mathbf{W}_{mm}\mathbf{z}_i) \tag{5.39}$$

$$\omega_{t,i} = \frac{\exp(\beta_{t,i})}{\sum_i \exp(\beta_{t,i})} \tag{5.40}$$

where, $\mathbf{q}$ and $\mathbf{W}_{*,*}$ are the parameters to be trained. In the decoding phase, we use the s2s-cell of H-LSTM model in the upper cell to estimate hidden vector $\mathbf{h}_t$ for predicting the next token.

5.4 Experiments

The experiments are conducted here not only to observe the performance of each LSTM variant discussed in Section 5.3 but also to compare them with the standard baselines. The effectiveness of the models is compared using several metrics and different model architectures.

5.4.1 Hindi Word2Vec Embeddings

Word2Vec based word embeddings is a good approach for representing text in natural language processing under a deep-learning paradigm as it provides a better result than one-hot encodings [218]. Word-embeddings represent each token[5] in the vocabulary with a numeric vector of a certain length. A simple Word2Vec[6] approach with the skip-gram model is used to learn the word-embeddings through a shallow neural network. A small corpus consisting of entire delexicalised training data and 10K Hindi-monolingual[7] [293] sentences are used to train the word-embeddings. The training parameters of a window and vector size are empirically set to 5 and 100, respectively.

5.4.2 Evaluation Metrics and Baseline-Models

For evaluating the NLDG systems, researchers have generally considered both objective as well as human evaluation. We have compared the models on several standard evaluation metrics: BLEU-score [274], T-Error (Total Error), S-Error (Slot Error) and BV-Error (Binary-Value Error) [189]. In general, T-Error denotes all miss-matches of slot-values in a DA and the corresponding generated utterance, while S-Error considers miss-matches only for those slots which do not have binary-values. BV-Error denotes miss-matches of those slots which have binary-values. All the error metrics were calculated by averaging mismatched errors over each of the realisations in the entire corpus. These models are compared with the standard baselines models listed below:

[5] A word belongs to Delexicalised reference utterances of the data.
[6] https://radimrehurek.com/gensim/index.html
[7] A set of 10K senten are extracted randomly form http://www.cfilt.iitb.ac.in/iitb_parallel/.

- Rule-based (HDC) generator,
- K-Nearest Neighbors (KNN) based generator,
- n-gram for class-based generator proposed by [27].

5.4.3 Datasets

The generation-systems are targeted towards building a spoken dialogue system capable for providing details of the desired restaurant in Allahabad[8] (a city in India). We considered eight DA-types, i.e. *inform*: presenting required information about restaurants to the user, *confirm*: to confirm about a slot value conveyed by the user. The restaurant domain consists of 8 attributes (slots) like *food*, *area*, *price range*. The detailed ontology is presented in Section 2.1.3.

The corpus is collected in the domain of searching a restaurant in a city by a group of Hindi-speaking people. All the persons were first shown a set of pairs of simple DA and corresponding realisation to get the idea of the task. Afterwards, each participant has been presented with an unseen DA comprised of act type followed by a set of slot-value pairs and asked to input an appropriate natural language sentence in Hindi. We managed to collect **3K** pairs of DAs and corresponding utterances over $\sim$200 distinct DAs.

5.4.4 Experimental Setups

The RNNLG framework based generators were implemented using the PyDial Framework[9] in Theano-Library. The models are trained on an individual corpus partitioned in the ratio of 3:1:1 of training, validation and testing set with stochastic gradient-descent and back-propagation. L2-regularisation is applied in order to prevent over-fitting with regularisation factor 10^{-7}. Additionally, early stopping criteria based on the validation error has also been incorporated to avoid over-fitting.

All the RNNLG framework based models are trained using a cross-entropy loss function, estimated between the predicted word distribution $\mathbf{p}_t$ and the actual word distribution $\mathbf{y}_t$, including the regularised DA-vector and its transition dynamics as in Equation 5.41:

$$\mathcal{L}(\theta) = \sum_t \mathbf{p}_t^T \log(\mathbf{y}_t) + ||\mathbf{d}_T|| + \sum_{t=0}^{T-1} \eta \xi^{||\mathbf{d}_{t+1} - \mathbf{d}_t||} \tag{5.41}$$

Where, $\mathcal{L}(\theta)$ corresponds to cross-entropy loss, θ is training weight-matrix, $\mathbf{d}_T$ is DA-vector of the previous time-step. η and ξ are regularisation constants set to 10^{-4} and 100 respectively.

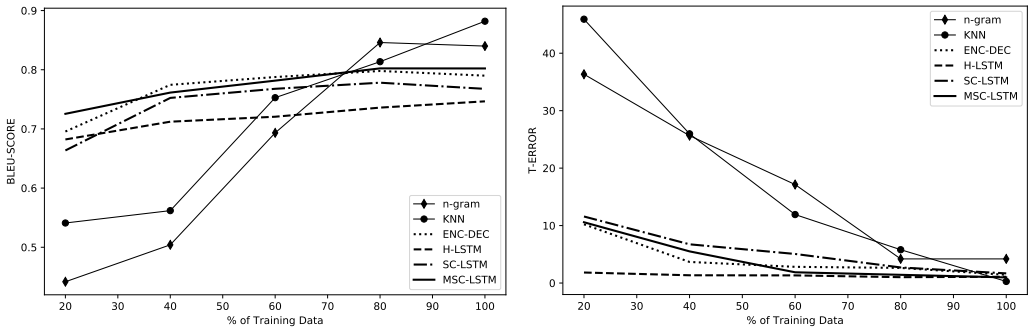
[8] https://en.wikipedia.org/wiki/Allahabad

[9] The Cambridge University Python Multi-domain Statistical Dialogue System Toolkit http://www.camdial.org/pydial/.

In the decoding phase, the model generates 20 utterances based on the beam-search decoding from which the top 5 are selected based on *T-Error*[10][294]. The results depict the top performance of the models as obtained in the experiments.

Table 5.1 Result of various Models. (Underlined-Models are the baseline models. Errors are in percentage(%).)

Models	Test			Validation		
	BLEU	**T-Error**	**S-Error**	**BLEU**	**T-Error**	**S-Error**
HDC	0.26	0.00	0.00	-	-	-
n**-gram**	0.85	4.20	2.57	-	-	-
KNN	0.88	0.28	0.24	-	-	-
V-LSTM	0.68	2.36	1.38	0.71	1.37	1.86
H-RNN	0.69	2.21	1.21	0.72	1.32	0.54
SC-RNN	0.70	1.67	1.20	0.74	1.88	1.06
ENC-DEC	0.79	1.43	0.83	0.77	3.01	2.06
H-LSTM	0.75	1.09	**0.19**	0.76	1.35	**0.39**
SC-LSTM	0.77	1.68	1.16	0.77	1.65	0.83
MSC-LSTM	**0.80**	**0.98**	0.59	**0.79**	**1.16**	0.77



(a) BLEU-Score

(b) Total Error

(c) Slot Error

(d) Binary-Value Error

Figure 5.7 Comparison of RNNLG Models on (%) of Training-Data.

5.5 Results & Analysis

In this section, we compare the output of all the models discussed in section 5.3. The comparison of test and validation results is shown in Table 5.1. The HDC baseline model generates error-free utterances in terms of

[10]Utterances having minimum Total Error (T-Error) are selected.

slot-value pairs but with the lowest BLEU-score. The reason is that HDC is designed based on the pre-defined rules to generate rigid utterances which are different from the human collected utterances. Another drawback of the HDC model is scalability making this model difficult to expand over the large domains. Next, the n-gram baseline model shows improvement on BLEU-score compared to HDC but render the worst S-Error due to missing slot-value pairs in the output. The third baseline model, KNN, which is based on the similarity of dialogue-acts of testing data to training and validating part of the corpus, shows the best results in terms of BLEU-score and Error-values if 100% data is opted for training, testing and validation in the ratio of (3:1:1). But, if the training data size is reduced, its BLEU and Error-values get worse faster than the other models, as shown in Figure 5.7.

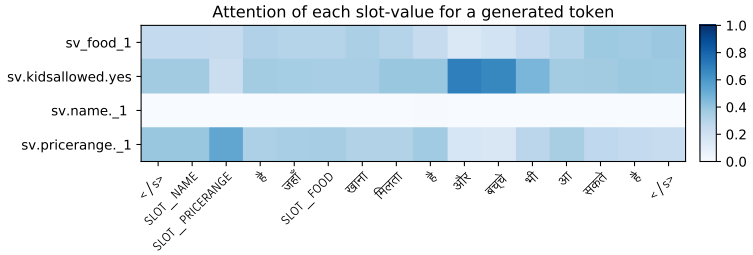


Figure 5.8 ENC-DEC: key-phrase processing sequence.

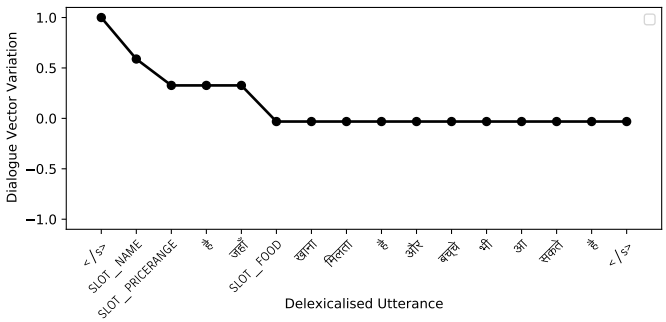
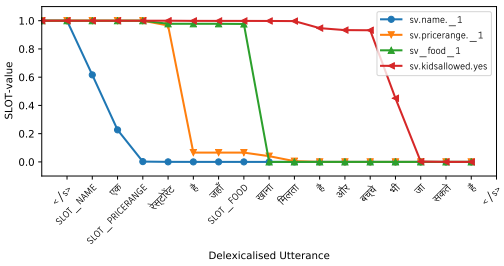
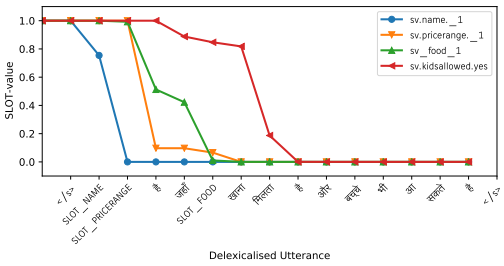


Figure 5.9 H-LSTM: key-phrase processing sequence.

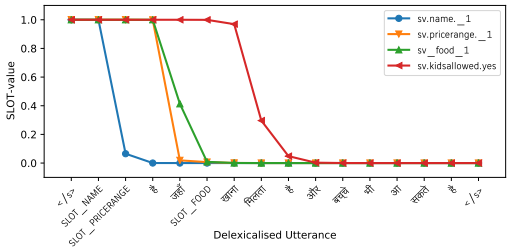
The RNNLG models have not only shown greater performance than the above models but also found to be scalable and adaptable towards the large domain. However, the ENC-DEC result is not up to the mark as compared to heuristically-gated and semantically-controlled models. This is due to the inherent limitation of attention-mechanism, which does not prevent the slot repetitions in the generation process, as shown in Figure 5.8. It signifies that the attention mechanism in END-DEC is unable to model the DA information well. This limitation is overcome in the later models by checking the slot repetition by modelling the DA-information separately through a DA-cell. This fact is evident in Figure 5.7. The key-phrase processing sequence of all the models are constructed corresponding to a DA = 'inform(name="गोल्डन रसोई रेस्तरां", price range="महंगा", food="चाइनीस", kidsall-owed = "yes")' which represent how various models process the given DA as depicted in Figures 5.8, 5.9 and 5.10.



(a) SC-RNN : key-phrase detector

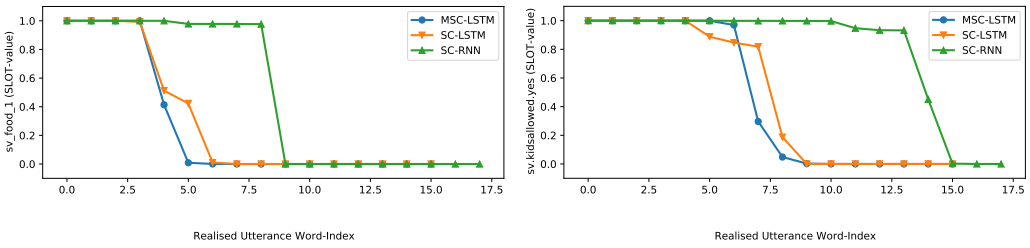


(b) SC-LSTM : key-phrase detector



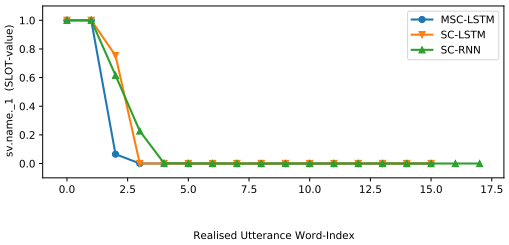
(c) MSC-LSTM : key-phrase detector

Figure 5.10 Key-phrase detection in Semantically-conditioned Models.

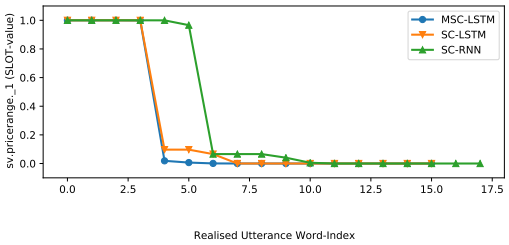


(a) Change in food-vector

(b) Change in kidsallowed-vector



(c) Change in name-vector



(d) Change in pricerange-vector

Figure 5.11 Change of SV-vector during the Transformation.

Table 5.2 BLEU-score of Models on various scales of the data (%).

Models	**20%**	**40%**	**60%**	**80%**	**100%**
HDC	0.21	0.23	0.24	0.25	0.26
n-gram	0.44	0.50	0.69	0.85	0.85
KNN	0.54	0.56	0.75	0.81	0.88
V-LSTM	0.56	0.59	0.62	0.64	0.68
H-RNN	0.59	0.61	0.67	0.63	0.69
SC-RNN	0.63	0.66	0.69	0.66	0.70
ENC-DEC	0.70	**0.77**	0.78	**0.80**	0.79
H-LSTM	0.68	0.71	0.72	0.74	0.75
SC-LSTM	0.66	0.75	0.77	0.78	0.77
MSC-LSTM	**0.73**	0.76	**0.79**	**0.80**	**0.80**

Higher is better.

In the category of neural NLDG models, V-LSTM presents comparatively lowest performance as it does not model the DA-cell separately. On the contrary, the RNN based models, H-RNN and SC-RNN show better performance than the V-LSTM model as they do have the DA-cell, but their accuracy is lower than their corresponding LSTM models, i.e. H-LSTM and SC-LSTM. It proves the supremacy of LSTM models over the RNN as the former is better able to handle the problem of vanishing gradient in the generation of long-length utterances when associated with an external DA-cell in the network.

Table 5.3 Total Error of Models on various scales of the data (%).

Models	**20%**	**40%**	**60%**	**80%**	**100%**
HDC	0.00	0.00	0.00	0.00	0.00
n-gram	36.34	25.65	17.14	4.20	4.20
KNN	45.92	25.97	11.92	5.8	0.28
V-LSTM	22.98	11.54	9.06	3.96	2.36
H-RNN	18.70	9.35	6.37	2.58	2.21
SC-RNN	16.33	7.54	3.99	1.76	1.67
ENC-DEC	10.22	3.67	2.82	2.63	1.43
H-LSTM	**1.83**	**1.35**	**1.33**	**1.02**	1.09
SC-LSTM	11.58	6.74	5.06	2.71	1.68
MSC-LSTM	10.58	5.51	1.87	1.43	**0.98**

Lower is better.

Table 5.4 Slot Error of Models on various scales of the data (%).

Models	**20%**	**40%**	**60%**	**80%**	**100%**
HDC	0.00	0.00	0.00	0.00	0.00
n-gram	23.87	18.46	10.84	2.57	2.57
KNN	42.03	23.39	10.11	4.77	0.24
V-LSTM	12.39	6.68	3.38	2.38	1.38
H-RNN	3.28	3.19	2.49	1.86	1.21
SC-RNN	8.94	3.88	1.88	1.37	1.20
ENC-DEC	8.63	2.02	1.72	2.08	0.83
H-LSTM	**0.48**	**0.19**	**0.19**	**0.19**	**0.19**
SC-LSTM	8.77	5.60	3.67	1.75	1.16
MSC-LSTM	7.33	3.97	1.26	0.96	0.59

Lower is better.

Table 5.5 Binary-Value Error of Models on various scales of the data (%).

Models	**20%**	**40%**	**60%**	**80%**	**100%**
HDC	0.00	0.00	0.00	0.00	0.00
n-gram	12.47	7.19	6.3	1.63	1.63
KNN	3.89	2.58	1.81	1.03	0.04
V-LSTM	10.59	4.86	5.68	1.58	0.98
H-RNN	15.42	6.16	3.88	0.72	1.00
SC-RNN	7.39	3.66	2.11	0.69	0.47
ENC-DEC	1.59	1.65	1.10	0.55	0.60
H-LSTM	**1.35**	**1.14**	1.14	0.83	0.90
SC-LSTM	2.81	1.16	1.39	0.96	0.52
MSC-LSTM	3.25	1.54	**0.61**	**0.47**	**0.39**

Lower is better.

While the heuristically-gated model has an advantage over semantically-controlled mechanism in terms of Slot Error when the sentences are fully and properly delexicalised, as in Figure 5.7c. This is because the

semantically-controlled models work as a key-phrase detector as they map the slot-value as key to its corresponding generated output-phrase with the help of an additional trainable reading-gate (Figure 5.10c). In contrast, the heuristically-gated model aligns the input DA to its associated phrase only for the slot-value pair that are delexicalised, making it unsuitable for binary-value slot-value, as shown in Figure 5.7d.

On comparing the semantically-controlled models, the MSC-LSTM updates the SV-vector temporally faster than the SC-LSTM and SC-RNN during the generation, as shown in Figure 5.11. The reason why MSC-LSTM delivers better performance than the SC-LSTM, is because the former incorporates the influence of DA-vector $\mathbf{d}_t$ not only in the reading-gate but also in input-, forget- and output-gates during the training (see its model architecture in Figure 5.5).

Additionally, the BLEU-score, T-Error, S-Error and BV-Error of all the NLDG models are presented in Table 5.2, 5.3, 5.4 and 5.5, corresponding to various scales of the training data, e.g. 20%, 40%, 60%, 80% and 100%.

Example dialogue acts and their top-5 realisations of top performing models are shown in Table-5.6.

5.6 Summary

In this chapter, we explore and discuss the Natural Language Dialogue Generation (NLDG) component in the SDS pipeline for building a Hindi conversational system. Like the language understanding task, it also requires a corpus to build an NLDG system, and there are no available such corpora in an Indic language. Hence, the chapter has first offered a corpus consisting of pairs of dialogue-act and natural sentences without any semantic alignment annotations suitable for learning corpus-based response generations models. The chapter has also investigated the corpus-based models aiming at learning response generation directly from the data. Three baseline approaches have been experimented: the hand-crafted (template) based generator (HDC), the class-based n-gram language generator and the example-based KNN approach.

Since we are focussing on building a dialogue system scalable to bigger domains, the chapter has explored Recurrent Neural Network (RNN) and other deep-learning paradigms for converting the dialogue act to a natural response. We begin with a discussion about neural network training, from back-propagation, gradient computation to stochastic gradient descent. This RNNLG framework has been adapted to explore and construct RNN models different capabilities, i.e. (a) heuristically-gated models (H-RNN, H-LSTM), (b) semantically-controlled models (SC-RNN, SC-LSTM, MSC-LSTM) and (c) ENC-DEC, for generating responses for a Hindi dialogue system. The general architecture in the RNNLG framework follows a combined process of sentence-planning and surface-realisation by a recurrent structure. The sentence-planning part is formulated by the Dialogue-Act modelling layer (DA-cell), while the actual surface-realisation (construction of a natural sentence) is obtained by a sequence-to-sequence layer (s2s-cell).

Table 5.6 Samples of top 5 realisations of top RNNLG-models.

1.	**inform(name=**पिंड बलूची**;area=**सिविल लाइंस**;pricerange=**मध्यम**;kidsallowed=no)**
#	**MSC-LSTM**
	पिंड बलूची जो सिविल लाइंस में मध्यम एवं अच्छा भोजनालय है एवं यहाँ बच्चों को आने की अनुमति नहीं है। पिंड बलूची जो सिविल लाइंस में मध्यम एवं अच्छा भोजनालय है जहाँ बच्चों का जाना मना है। पिंड बलूची जो सिविल लाइंस में मध्यम एवं अच्छा रेस्टोरेंट है एवं यहाँ बच्चों को आने की अनुमति नहीं है। पिंड बलूची जो सिविल लाइंस में मध्यम एवं अच्छा भोजनालय है जहाँ बच्चों को आने की अनुमति नहीं है। पिंड बलूची जो सिविल लाइंस में मध्यम एवं अच्छा रेस्टोरेंट है जहाँ बच्चों का जाना मना है।
#	**SC-LSTM**
	पिंड बलूची सिविल लाइंस में मध्यम भोजनालय का रेस्टोरेंट है जहाँ बच्चों को आने की अनुमति नहीं है। पिंड बलूची सिविल लाइंस में मध्यम भोजनालय का भोजनालय है जहाँ बच्चों को आने की अनुमति नहीं है। पिंड बलूची सिविल लाइंस में मध्यम भोजनालय का रेस्टोरेंट है जहाँ बच्चों को प्रवेश की अनुमति नहीं है। पिंड बलूची सिविल लाइंस में मध्यम भोजनालय का भोजनालय है जहाँ बच्चों को प्रवेश की अनुमति नहीं है। पिंड बलूची सिविल लाइंस में मध्यम भोजनालय का भोजनालय है जहाँ बच्चे नहीं जा सकते हैं।
#	**H-LSTM**
	पिंड बलूची जो सिविल लाइंस में है , यहाँ मध्यम खाना मिलता है एवं वहाँ बच्चों को आने की अनुमति नहीं है। पिंड बलूची जो सिविल लाइंस में है , यहाँ मध्यम खाना मिलता है एवं यहाँ बच्चों को आने की अनुमति नहीं है। पिंड बलूची जो सिविल लाइंस में है , यहाँ मध्यम खाना मिलता है परन्तु वहाँ बच्चों को आने की अनुमति नहीं है। पिंड बलूची जो सिविल लाइंस में है , यहाँ मध्यम खाना मिलता है। पिंड बलूची जो सिविल लाइंस में है , यहाँ मध्यम खाना मिलता है एवं वहाँ बच्चों का जाना मना है।
#	**ENC-DEC**
	पिंड बलूची सिविल लाइंस में स्थित एक मध्यम भोजनालय है जहाँ बच्चों का जाना मना नहीं है। पिंड बलूची सिविल लाइंस में स्थित एक मध्यम भोजनालय है जहाँ बच्चों का आना मना नहीं है। पिंड बलूची सिविल लाइंस में स्थित एक मध्यम भोजनालय है जहाँ बच्चों का जाना मना है। पिंड बलूची सिविल लाइंस में स्थित एक मध्यम भोजनालय है जहाँ बच्चों को आने की अनुमति नहीं है। पिंड बलूची सिविल लाइंस में स्थित एक मध्यम भोजनालय है जहाँ बच्चों के लिए अनुमति नहीं है।
2.	**inform(name=**रजवाड़ा**;area=**कर्नल गंज**;food=**इंटरनेशनल**;kidsallowed=yes)**
#	**MSC-LSTM**
	रजवाड़ा कर्नल गंज में स्थित है जहाँ इंटरनेशनल खाना मिलता है और बच्चों को प्रवेश की अनुमति है। रजवाड़ा कर्नल गंज में स्थित है जहाँ इंटरनेशनल खाना मिलता है और बच्चे भी जा सकते हैं। रजवाड़ा कर्नल गंज में स्थित है , इंटरनेशनल खाना मिलता है और बच्चों को प्रवेश की अनुमति है। रजवाड़ा कर्नल गंज में स्थित है , इंटरनेशनल खाना मिलता है और बच्चे भी जा सकते हैं। रजवाड़ा कर्नल गंज में स्थित है जहाँ इंटरनेशनल खाना मिलता है और बच्चे बच्चे जा सकते हैं।
#	**SC-LSTM**
	रजवाड़ा कर्नल गंज में स्थित है जहाँ इंटरनेशनल खाना मिलता है और बच्चे भी जा सकते हैं। रजवाड़ा कर्नल गंज में स्थित है , में इंटरनेशनल खाना मिलता है और बच्चे भी जा सकते हैं। रजवाड़ा कर्नल गंज में स्थित है जहाँ इंटरनेशनल खाना मिलता है और वहाँ बच्चे भी जा सकते हैं। रजवाड़ा कर्नल गंज में स्थित है , में इंटरनेशनल खाना मिलता है और वहाँ बच्चे भी आ सकते हैं। रजवाड़ा कर्नल गंज में स्थित है , में इंटरनेशनल खाना मिलता है और बच्चे भी आ सकते हैं।
#	**H-LSTM**
	कर्नल गंज क्षेत्र में रजवाड़ा नाम का एक भोजनालय है जहाँ इंटरनेशनल खाना मिलता है और बच्चों को प्रवेश की अनुमति है। कर्नल गंज क्षेत्र में रजवाड़ा नाम का एक भोजनालय है जहाँ इंटरनेशनल खाना मिलता है और बच्चों का आना मना नहीं है। कर्नल गंज क्षेत्र में रजवाड़ा नाम का एक भोजनालय है जहाँ इंटरनेशनल खाना मिलता है और बच्चों को यहां की अनुमति है। कर्नल गंज क्षेत्र में रजवाड़ा नाम का एक भोजनालय है जहाँ इंटरनेशनल खाना मिलता है और बच्चों को आने की अनुमति है। कर्नल गंज क्षेत्र में रजवाड़ा नाम का एक भोजनालय है जहाँ इंटरनेशनल खाना मिलता है और बच्चों को प्रवेश की अनुमति है।
#	**ENC-DEC**
	रजवाड़ा कर्नल गंज में स्थित है जहाँ इंटरनेशनल खाना मिलता है और बच्चों को प्रवेश की अनुमति है। रजवाड़ा कर्नल गंज में स्थित है जहाँ इंटरनेशनल खाना मिलता है और बच्चे भी जा सकते हैं। रजवाड़ा कर्नल गंज में स्थित है , यहाँ इंटरनेशनल भोजन मिलता है। यहाँ पर बच्चों का आना मना नहीं है। यहाँ पर बच्चों का आना मना नहीं है। रजवाड़ा कर्नल गंज में स्थित है , यहाँ इंटरनेशनल भोजन मिलता है। यहाँ पर बच्चों का आना मना नहीं है। यहाँ पर बच्चों का आना मना है। रजवाड़ा कर्नल गंज में स्थित है जहाँ इंटरनेशनल खाना मिलता है और वहाँ बच्चे जा सकते हैं।

The chapter compares the performance of all the generation methods on BLEU-score, Total Error (T-Error) and Slot Error (S-Error). The HDC approach is the most robust in terms of slot error rate simply because the rendering of information is strictly managed by the template rules. However, such way of generation not only sacrifices the improvisation in human language and generates surface forms that are much more rigid and stylised

than the other approaches. When comparing the RNNLG models to the baselines, the following conclusions can be drawn:

1. The RNNLG models outperform all the baseline models, which shows the distributed representation and the modelling of long-term dependencies using LSTM.

2. Semantically conditioned models (e.g. SC-LSTM, MSC-LSTM) are considered top choices due to their learnable controlling gates. In contrast, Heuristically gated models (e.g. H-LSTM) are suitable for examples where the utterances are fully delexicalisable and the amount of data is relatively insufficient.

3. Our, MSC-LSTM (Modified SC-LSTM) is the best performing model as it can remember key-phrases corresponding to DA-type and slot-value pairs by incorporating DA in a different way than the SC-LSTM architecture.

Therefore, the proposed MSC-LSTM can be considered a more suitable model for the NLG tasks. The next chapter discusses various ways to transform the natural language to a synthesised speech as well as evaluate their quality.

Chapter 6

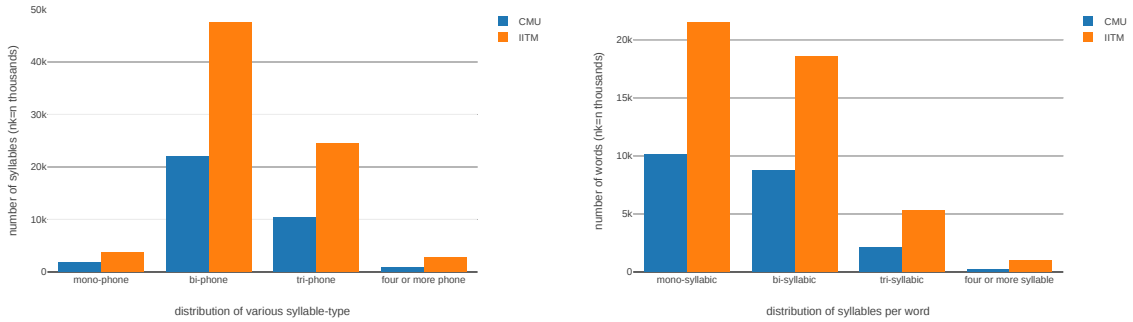
Quality Assessment of Synthesised Speech

6.1 Introduction

This chapter presents a framework LBOE that investigates the synthetic speech quality at two levels: First, *perceptually salient acoustic-features* are identified which define the perceptual quality space of a synthetic speech holistically. Second, *quality-prediction models* are constructed using the perceptually salient acoustic-features to estimate perceptual quality-rating in a non-intrusive manner as a black-box approach. The main goal of the research is to propose a novel framework that explores the *generalisation capabilities* of low-level descriptor-based perceptual features and investigates to what extent they can be used to measure the synthetic speech quality at all without subjective testing. Nonetheless, the synthetic speech quality can not be only attributed to a number of perceptual characteristics; the aspects of model (TTS) type and robustness need to be explored too. A special case of Non-intrusive Quality Assessment (NiQA), Leave-One-Model-Out (LOMO), is discussed to show the effectiveness of the proposed framework [295]. Such frameworks are not thoroughly explored previously in the area of TTS quality evaluation. We have also compared the evaluation results with other NiQA models, e.g. Quality-Net [180] and MOSNet [181], to show the performance of our framework.

The remainder of this chapter is organised as follows: Section 6.2 describes the speech material used in the study, with analysing the properties of datasets in Section 6.2.1 to be used in building the TTS models in Section 6.2.2. Conventional evaluation methods of synthesised speech, i.e., Subjective and Objective Evaluation, are briefly discussed in Section 6.3. The proposed framework of Learning-Based Objective Evaluation is discussed in Section 6.4, comprising the subtopics of dataset preparation, feature extraction, selection & normalisation, classification and quality prediction. A brief analysis and comparison of all the evaluation methods are elaborated in the Results & Analysis, Section 6.5. Relevant aspects of the work done are discussed in Section 6.6, and conclusions are drawn in Section 6.7.

6.2 Speech material



(a) Distributions of phones per syllable

(b) Distributions of syllables per word

Figure 6.1 Distribution of number of phones per syllable and number of syllables per word for both IITM and CMU datasets.

6.2.1 Dataset Description

We have used two different open-source Hindi-TTS datasets: CMU-Indic Hindi TTS dataset [296] and IITM Hindi speech dataset [297]. Both the datasets are collected in female voice on a set of Hindi sentences. Statistical comparison for both the datasets is shown in Table 6.1. On all the properties, IITM dataset is superior to CMU dataset. IITM dataset covers more unique words, unique syllables and unique phones than CMU datasets, as it comprised more utterances in the corpus. The average length of a sentence is also greater for IITM than the CMU dataset.

In terms of phones, IITM dataset does not contain phone '/e/' ['/ए/'][1] while CMU dataset lacks phone '/rx/'['/ऱ/'], '/nk/'[Hindi nukta] [299], '/ng/'['/ङ/']. Otherwise, both the datasets share almost all sorts of phonemes in a set of phonetically rich utterances.

Both datasets are analysed based on the distribution of syllables on their constituent phone and words on syllables. It's clearly observed from Figure 6.1a that the syllables composed of two phones occur more frequently across both the datasets mainly of CV[2] form, compared to VC. Tri-phone syllables of the CVC form are next most frequent. Syllables with a single phone (V structure) or four or more phones have very low occurrences.

Words, in the datasets, are also analysed under categories of mono-syllabic, bi-syllabic, tri-syllabic etc. for both datasets. Distribution in Figure 6.1b shows the descending number of words with the increase in constituent syllables per word [300]. Overall, both the datasets are phonetically rich and balanced, covering most of the phonetic properties of the Hindi language, but IITM has more content than CMU and should be able to train

[1] CMU phone-set for indic languages [298]
[2] C:Consonant, V:Vowel

Table 6.1 Details of CMU and IITM Hindi TTS-Datasets.

Properties	**CMU**	**IITM**
Total Words	21223	46341
Average words per sentence	17.68	20.00
Total Sentences	1793	2318
Total Syllables	34477	123037
Total Phones	78825	180125
Unique Words	4214	9498
Unique Syllables	1990	4203
Unique Phones	56	58
Duration (H:M:S)	2:50:44	5:11:6

a more natural TTS system. Later it is empirically shown that category-wise each model trained on IITM performed better than CMU dataset.

6.2.2 Building Hindi TTS systems

For the current study, we aim to cover leading TTS technologies as used in research as well as state-of-the-art commercial systems. Both TTS datasets are used to build four types of unmodified "off-the-shelf" TTS systems: Unit selection speech synthesis (USS), Hidden Markov Model speech synthesis (HMM), Clustergen speech synthesis (CLU) and Deep Neural Network-based speech synthesis (DNN).

Unit Selection speech Synthesis (USS)

The Unit selection speech synthesis (USS) is fundamentally a cluster-based technique, which combines units of similar type (e.g. phones, diphones, syllables etc.) based on their acoustic differences [34]. The clusters are then indexed based on high-level features such as phonetic and prosodic context. However, its use in the embedded systems gets affected by their computational processing power and memory footprint. It is necessary to find a favourable compromise between the size of the speech corpus and the computational complexity of the unit-selection method [142].

Unit selection speech synthesis (USS) process get initiated with collecting all possible units from the speech database and ends on learning weights to find out unit cost (w_j^t) as well as target cost (w_j^c) of a particular phonetic unit to be fitted in a sequence in order to generate a meaningful utterance sound [34]. The input to the core unit selection block has two inputs: the generated utterance from the text processing part in the form of phonetic units and the speech corpus. The output would be a list of sound units selected from the speech corpus, which would be given to the final sound production system.

The above black-box can be understood as a process to reconstruct the input utterance acoustically by joining the units from the corpus. The output of this step of the unit selection process is an ordered list of corpus units

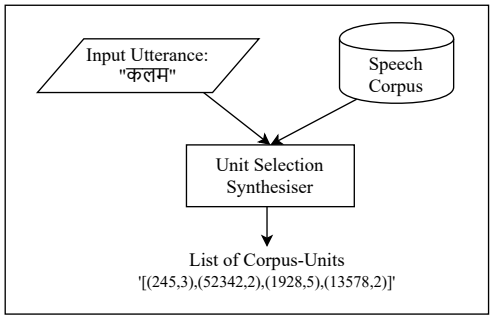


Figure 6.2 Abstract view of unit selection process.

as shown in the figure represented by a list of 2-valued tuples $[(index, size)]$. These are the units required to be concatenated to reproduce the reconstructed sound.

In this work, MaryTTS[3], an open-source tool, is used to build USS models on CMU and IITM datasets [301, 143]. Phone and Half-phone based contextual feature weights, considered as a base of units, are used for training and selection [144].

Two primary requirements of the USS are to structure corpus units and efficiently explore them. As the structure of sound units is represented as an ordered relationship between them and cost of occurrence in pair, the problem becomes an optimum-cost pathfinding problem in a directed weighted graph. The widely $G(V, A, C)$ used structure to represent all the units is proposed as finite-state modelling where V is a set of states representing possible states while exploring the target sequence, A arcs symbolise the corpus units, and each arc has a weight corresponding to the cost C of selecting that unit in the context of previous and next state in the neighbourhood.

To show the process in a more formal and intuitive way, Let's assume $T = (d_1, d_2, ..., d_N)$ is a target sequence of syllables where d_k is the k^{th} syllable in the sequence of N units. Suppose the set of corresponding matching candidate syllables from the corpus is denoted as $\Psi_k = \left\{d_k^1, d_k^2, ..., d_k^{M_k}\right\}$ where M_k represents all the candidate syllables in the corpus that matched with the target syllable $\hat{d_k}$. Further, with $i, j \in [\![1 : N]\!], i < j$ a sequence of matching target syllable i to j is denoted by $U_{i,j}^x$ where x means x^{th} matching of target sequence from d_i to d_j. With all these assumptions, we define a set of all corpus units as $\Omega_{i,j} = \left\{U_{i,j}^1, U_{i,j}^2, ..., U_{i,j}^{M_k}\right\}$ that matched with target sequence from d_i to d_j.

The above description helps understand the structural details of sound units in the corpus and a glimpse of how a given target sequence is matched with them as an optimum pathfinding problem. It is Formalised with $1 \leq h < i$ as the following cost function:

$$U^* = \underset{U=U_{1,h}^{w_{1,h}},...,U_{j,N}^{w_{j,N}}}{\arg\min} \left(C\left(U_{1,h}^{w_{1,h}}, U_{h,i}^{w_{h,i}}, ..., U_{j,N}^{w_{j,N}}\right)\right) \tag{6.1}$$

[3]The MARY Text-to-Speech System (MaryTTS) http://mary.dfki.de/

It is done through the Viterbi algorithm to get the most accurate sequence of sound units from the corpus through estimating the cost of each unit determined by two cost functions target cost (C_t) and concatenation cost (C_c). Consequently, a unified cost function is represented as $C\big(U_{i,j}^{w_{i,j}}\big) = C_t\big(U_{i,j}^{w_{i,j}}\big) + C_c\big(U_{h,i}^{w_{h,i}}, U_{i,j}^{w_{i,j}}\big)$ where $U_{h,i}^{w_{h,i}}$ is the predecessor of $U_{i,j}^{w_{i,j}}$ in the candidate sequence. Then the optimisation equation becomes:

$$U^* = \underset{U=U_{1,h}^{w_{1,h}},\ldots,U_{j,N}^{w_{j,N}}}{\arg\min} \Big(W_{tc} \sum_{U} C_t\big(U_{i,j}^{w_{i,j}}\big) + W_{cc} \sum_{U} C_c\big(U_{h,i}^{w_{h,i}}, U_{i,j}^{w_{i,j}}\big) \Big) \tag{6.2}$$

The central issue in the entire unit selection process is the estimation of the cost function weights W_{tc} and W_{cc}. Regression is the most widely used method for this. It learns the weights for both concatenation cost as well as target cost separately. Earlier experiments have shown that a combination of cepstral distance, pitch difference and difference in power at the concatenation point has a significant characteristic to resemble the perceptual quality. Similarly, the target cost weights are also obtained through applying multiple linear regression on the objective distance function [34].

The unit selection algorithm generates the output simply a sequence of unit positions present in the corpus. So the final task of TTS is to chain up the units with minimum discontinuities (glitches) at the point of concatenation, and it can also implement some prosody (modification) adaptation, e.g. changing pitch or accelerating speech rate throughout the complete signal [302].

Hidden Markov Model speech synthesis (HMM)

The existing limitation of unit selection promotes the application of statistical parametric based approaches for speech synthesis. The USS has investigated the selection of sound units of optimal size. It has been observed that the larger the unit size, the longer the database is required to cover all possible units in the domain. On the other hand, for the smaller units, more joining points during the synthesis effects naturality of the sound [303]. However, the larger database to train a synthesiser may look an easy way to follow, but as databases grow in multiple tens of hours, handling time-dependent quality variations in speech become a challenging task. In addition, a very large database requires a much higher degree of computational resources, which hinder the unit selection based TTS systems to be incorporated in embedded devices or a variety of voices and languages.

In contrast, the issues mentioned above are the specific counterparts of statistical parametric synthesis; the Hidden Markov Model (HMM) based model is one of them [145]. The HMM-based TTS system works in two phases. The first is to extract temporal parameters, i.e. spectral (e.g., Mel-cepstral coefficients) and excitation features (e.g., log F0 and its dynamic features) from the speech database and then model them. We have built two separate models for CMU and IITM datasets. The second phase generates a sequence of desired speech parameters through trained models for a given word sequence to be synthesised. The parameters sequence

with maximum output probability is considered for forming the final sound wave [146]. Model parameters are estimated by maximising likelihood criteria as below:

$$\hat{\lambda} = \arg\max_{\lambda} \left\{ p(\mathbf{O}|\mathbf{W}, \lambda) \right\} \tag{6.3}$$

where λ denotes parameters the model will train on, $\mathbf{O}$ represents training data and $\mathbf{W}$ is a sequence of words corresponding to $\mathbf{O}$.

The second phase generates a sequence of desired speech parameters, $\mathbf{o}$, through the set of trained models $\hat{\lambda}$, for a given word sequence $\mathbf{w}$ to be synthesised. The same process is used during speech recognition but in reverse order. The parameters sequence $\hat{\mathbf{o}}$ with maximum output probability is considered for forming the final sound wave [146].

$$\hat{\mathbf{o}} = \arg\max_{\mathbf{o}} \left\{ p(\mathbf{o}|\mathbf{w}, \hat{\lambda}) \right\} \tag{6.4}$$

It has several advantages over USS and disadvantages too. Many advantages are related to its flexibility in handling different variations efficiently due to its parametric-statistical nature, which enables it to transform (adapt) voice characteristics, speaking styles, and emotions. USS required the control parameters to be tuned manually, but HMM does it automatically as it is based on well-defined statistical principles. Besides all its efficiencies, it has some significant drawbacks over USS as the output voice is not that natural.

Clustergen speech synthesis (CLU)

However, USS techniques have proud to rely on no or very little use of signal processing, still able to keep a hold on maintaining the crispness in the synthesised voice. It is possible only on the cost of building larger and larger datasets. In order to apply more stylistic variations, e.g. style, emotions, USS need more and more data in different styles to train on. Hence, the output model will also be bigger, make less usable everywhere.

HMM is not up to the mark in the domain of database-oriented approaches like USS. But it has some significant advantages too. It has the advantage of using smoothed data that enables it to cover and adapt many phonetic variations for synthesis, which does not require a very large database.

This section is about the use of Clustergen (CLU), a closer sibling of parametric TTS models, but it also has some characteristics of USS's as well like selecting a unit from a set (cluster of similar units) rather than based on the contextual cues [147]. It synthesises the speech for a given text using a trained CLU model.

As a general TTS, CLU also requires a set of pairwise spoken utterances and text transcription. Both text corresponding utterance is processed parallel to achieve final pair of units which required to be clustered for each

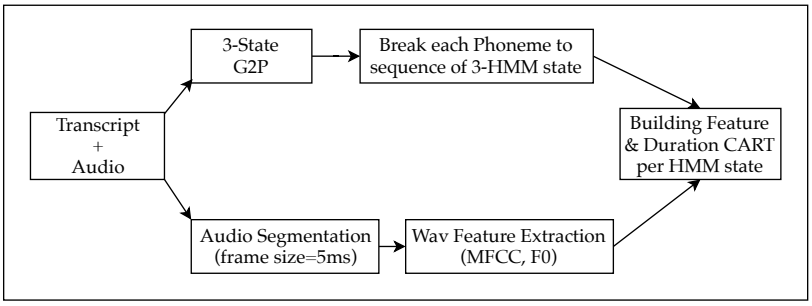


Figure 6.3 Training process of Clustergen (CLU) algorithm.

possible HMM-state obtained through automatic-labelling[4] [304]. Each extracted phone from the text mapped to 3 HMM-state labels that finally aligns with the voice.

The fixed frame size of 5 ms is used to segment the speech wave for extracting $F0$ and $MFCC$ feature vectors[5] [148]. In total, a vector of 25 features comprised of 24 $MFCC$s and a $F0$ is estimated for every 5ms and considered as a CLUNIT. This is a low-level representation of a speech frame which is supposed to be converted into high-level features such as preceding and following phonetic context in terms of both phonetic identity and phonetic features, prosodic context (pitch, duration of both preceding and following units), stress, syllable structure, word position.

Clustering CLUNITS is done through CART tree builder[6]. It works with finding a splitting criterion (a question on a particular feature) that split it into some clusters in order to minimise the impurity. Entropy is used here to estimate the impurity of a group of units:

$$E = \sum_{(x \in class)} prob(x) * log(prob(x)) \tag{6.5}$$

where, x is a CLUNITS in the cluster. Similarly, a CART tree is built for all vectors that belong to the same HMM-state. Additionally, the duration CART trees are also built per HMM-state independently to model durational variation.

The synthesis process starts with converting the input text into a phone string, where each phone links further to three sub-phonetic HMM-states. These sub-phonetic units will be processed by respective duration-CART, and HMM-state CART combinely generate averaged track coefficients used to synthesise speech using MLSA filter [149, 305].

For each target HMM-state, we consider the CART of same unit type, ask questions to reach leaf representing appropriate cluster. Means from the vectors of a selected cluster is added to the target vector. In a similar way, duration CART is utilised to find the duration of corresponding target HMM-state.

[4]EHMM-labeller of FestVox does this task http://festvox.org/bsv/x3308.html
[5]Edinburgh Speech Tools Library http://www.cstr.ed.ac.uk/projects/speech_tools/
[6]Edinburgh Speech Tools Library http://www.cstr.ed.ac.uk/projects/speech_tools/

The above process predicts all HMM-state sequences over the given target smoothed by a simple 3-point moving average. MLSA (Mel Log Spectrum Approximation) filter reconstructs the feature vector to speech wave implemented through a simple linear transform from the mel-cepstrum coefficient generated earlier [306].

DNN based speech synthesis (DNN)

Recently, several Deep Neural Network (DNN) based autoregressive models for TTS have been proposed, such as WaveNet [36], Deep-Voice 1, 2 & 3 [150–152], Tacotron-1 [37] and Tacotron-2 [38] etc. We use Tacotron-2[7] to build a DNN based TTS, an end-to-end TTS system better at handling the missing spectral information. The model first predicts Mel-scale spectrograms from the character embeddings of Hindi letters through a sequence-to-sequence recurrent network, followed by a separate autoregressive model (WaveNet[8]) to turn it into a waveform. The intermediate features (80-dimensional audio spectrogram) computed on 12.5-millisecond frames are not only capable of capturing the pronunciation of the words but also various nuances of human speech, i.e. volume, intonation and tempo.

The Tacotron-2 speech synthesis system consists of two components: (1) a recurrent sequence-to-sequence spectrogram prediction network with attention that predicts a sequence of mel spectrogram frames from an input character sequence, and (2) a modified version of the WaveNet [36] to generate time-domain audio wave inverting the predicted mel spectrogram frames.

6.3 Subjective and Objective quality evaluation of the synthesised speech

This section discusses various subjective and objective evaluation methods in brief. To counter the limitations of subjective and objective evaluation methods, we further devise a novel way of Learning-Based Objective Evaluation that uses the acoustic LLD features to compare various TTS models by automatic classification and prediction in the next section.

6.3.1 Subjective Evaluation Experiment

To obtain subjective ratings, one hundred twenty post-graduate students of age (mean=24.6, SD=2.86) studying mostly Masters & PhD are chosen to participate in the experiment with initial training of the listening task. Hindi is the primary language (Mother Tongue) for all of the participants involved in the subjective evaluation. Each student is rewarded with course credits in proportion with the number of units evaluated.

A set of speech files are produced on the selected two hundred thirty sentences (belonging to IITM corpus) through all the TTS models trained on both CMU and IITM datasets. All such speech files, including human

[7] Tacotron-2: https://github.com/Rayhane-mamah/Tacotron-2
[8] WaveNet vocoder: https://github.com/r9y9/wavenet_vocoder

speech, are segregated into ten sets so that each group has speech files from all the models in equal proportion. Later, one set is randomly assigned to each participant to carry out the subjective evaluation task.

The participant starts the evaluation task with listening and evaluating each speech file separately in two phases. In the first phase, he/she has to write the content of each speech file in unconstrained condition on typos and homophones. Later, the listener has to rate each speech file on a questionnaire in the second phase.

User's input in the first phase, is used for the *intelligibility* subjective evaluation. Its main objective is to find the TTS model with higher intelligibility score. The intelligibility of a speech synthesiser is measured by calculating the word error rates (WER) of sentences typed by the users with the corresponding sentences in the original transcript [307]. Hence, it shows how well the listener is able to identify the words in a spoken utterance.

As the original transcripts are in Devanagari script, it is first transliterated[9] to Roman script. The original transliterated transcripts and typed sentences are then normalised to have similar vowel-types before determining the WER score using a simple word-based edit-distance method for each TTS model, including the original speech files.

Input to a questionnaire in the second phase provides the basis to estimate subjective-evaluation measures on comprehensibility, naturalness and prosody analysis. Comprehensibility measures how well the listener understands the meaning of the sentence in a speech [308], while naturalness denotes how smooth the flow between different words and sounds are presented with appropriate pauses in a speech [309]. Hence, a high natural speech should be closer to the way human pronounce that utterance. On the other hand, prosody analysis compares the expressiveness of speech synthesisers in terms of prosodic-parameters (i.e. F0, duration and energy) [310].

The questionnaire is prepared based on the modified MOS-X questions on which participants have to rate the speech quality on various points asked in several questions [311]. The MOS-X questions 9 and 12 to 15 represent behavioural feelings and confidence of the listener, which are not directly related to the properties of speech synthesisers. So, we discard them.

Based on the basic nature, the remaining ten questions are grouped in 3 categories (shown in Appendix E.1). Averaging questions 1 to 4, we obtain *comprehensibility measure*; questions 5 to 8 provides a measure of *naturalness*; questions 9 to 10 signify *prosody analysis*. All the questions are to be rated on a 7-point scale, where higher point shows listener with high satisfactory and vice-versa for lower points. A detailed empirical analysis of the subjective evaluation is presented in the result section 6.5.1.

[9] A transliteration tool for Indic-languages: https://github.com/libindic/indic-trans

6.3.2 Objective evaluation measures

In general, the objective evaluation measures are intrusive in nature and require original speech files to estimate how distorted the synthesised speech is. We have used various approaches of objective evaluation to investigate the performance of TTS models. Mel-Cepstral Distortion (MCD), F0 RMSE (Root-Mean-Square-Error) are the most common objective evaluation measures for speech synthesisers.

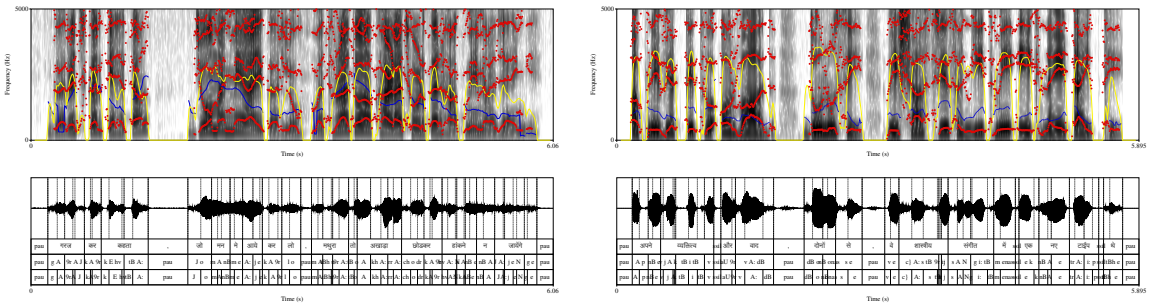
Some of the objective measures are also imported from speech coding area [312]. These objective measures computationally are of two types: time-domain based (e.g. signal-to-noise-ratio (SNR) measures) and frequency-domain based (e.g. LPC spectral distance) [313]. Most objective measures start with first segmenting the speech signal into a fixed frame size 25 ms, then computing the distortion measure between synthesised and original speech signals. The distortion measure of each speech frame is averaged out to find a single global measure of distorted speech through an intrusive way of objective speech evaluation. Note that objective measures do not conform to all properties of the standard distance metric. For this, these measures might not necessarily be symmetric, and some measures (e.g. log spectral) even yield negative results [313].

For the time and frequency domain objective measures, we have used multiple versions of signal-to-noise-ratio (SNR), i.e. Global SNR, Segmental SNR, frequency-weighted segmental SNR and BroadBand SNR [313, 314]. Spectral distance-based objective measures are estimated with Linear Predictive Coding Coefficients (LPCC) distortion and Log-Spectral Distortion. Itakura-Saito uses the LPC to measure gain-normalised spectral distortion between the LPC spectra of original and synthesised speech [315]. The Weighted Spectral Slope (WSS) is used to capture the distortion as weighted difference spectral slopes obtained again from the LPC. Cross-correlation is used to find the lag Difference (lagDiff) of original and synthesised speech in the time domain.

We have also evaluated the synthesis models on the Perceptually-Motivated distortion measures: BSD, PESQ; which takes account of the existing psychoacoustics knowledge of human perceptibility [316]. BSD measures are kind of perceptually-motivated measure which estimates distortion by calculating the difference between the loudness spectra of the original and the synthesised speech [317], while the PESQ uses a cognitive model to estimate MOS by calculating the difference between the internal representation of two signals.

6.4 Proposed Learning-Based Objective Evaluation (LBOE)

As the earlier evaluation techniques are incapable of providing a cost-effective and robust solution to the TTS quality assessment problem, we propose a novel Learning-Based Objective Evaluation approach to evaluate and compare TTS models on the basis of a predefined set of LLD's taken mostly from the energy, spectral, frequency and temporal features. We have carefully selected a minimal set of LLD features in order to evaluate the TTS



(a) TextGrid analysis (CMU-ORIG). (b) TextGrid analysis (IITM-ORIG).

Figure 6.4 Both sub-figures draw spectrum {formants(red), pitch (blue) and intensity (yellow)} first, then show the division of sound on words, syllables and phonemes boundaries.

models. We also observed that adding High-Level-Descriptors[10](HLD) based features does not improve the LBOE performance. This is seen during the comparison of the proposed feature-set with other feature-sets, i.e. IS13ComParE, AVEC13 etc., which includes both LLDs and HLDs. The strength and capability of the feature-set are statistically discussed in Section 6.5.2. Flow-chart of the proposed evaluation model is shown in Figure 6.5.

6.4.1 Data preparation

The first part of the framework is to prepare the speech data generated through various TTS systems. In order to observe the performance of the proposed feature-set at various levels of sound units, we split the speech files into *words*, *syllables* and *phonemes*. The procedure starts by using Festvox[11] to generate labels (*.*lab* files) of the synthesised speech for each TTS model, e.g. HMM, CLU, DNN, USS and ORIG[12], required for generating *TextGrid* files. Unified-parser[13] is used to extract the syllables from the Hindi transcripts [318]. The timestamp for splitting a speech into words, syllables and phonemes was written into TextGrid files with TextGridTools [14], which uses the pieces of information obtained from labels and unified-parser to split a speech file into words, syllables and phonemes (see Figure 6.4). Thus, in the end, we obtained three separate test sets with the word, syllables and phoneme level details.

6.4.2 Feature extraction & selection

For all sound files obtained at each level, acoustic LLD-based features are extracted using the openSmile toolkit[15] in the second part of the framework [319]. The selection of a minimal number of features not only avoided the problem of having too many features relative to the number of samples required for the evaluation but also

[10]Computed through the statistical functionals, i.e. mean and variances on LLDs as well as its derivatives.
[11]Building Indic Voices: http://festvox.org/bsv/x3528.html
[12]Here ORIG denotes the original human speech files.
[13]Unified Parser: https://www.iitm.ac.in/donlab/tts/unified.php
[14]TextGridTools https://textgridtools.readthedocs.io/en/stable/
[15]openSmile: audio feature extraction tool (https://audeering.github.io/opensmile/about.html)

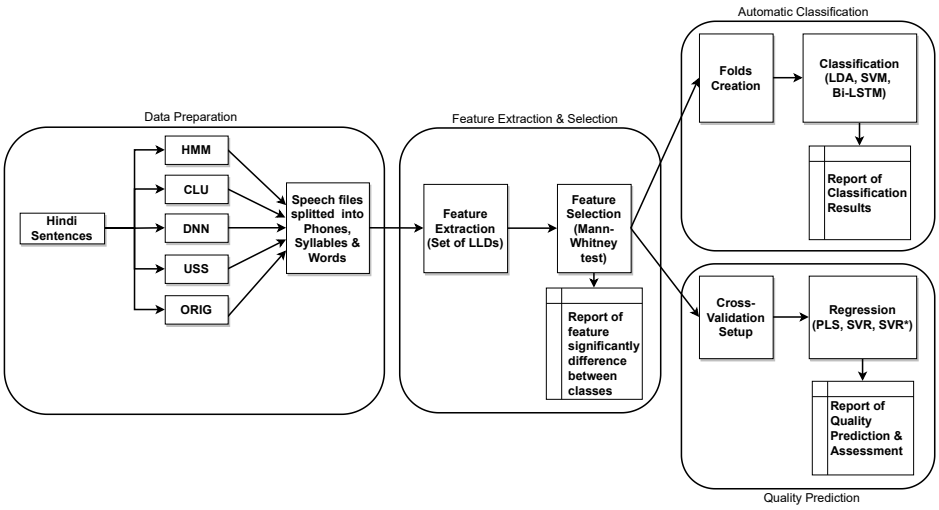


Figure 6.5 Experimental procedure of the LBOE, which has four parts: data preparation, feature extraction & selection and then automatic classification & quality prediction for the evaluation.

provided enough features to characterise prosody variations, i.e. duration, intensity and intonation in a speech. Good classification results can be obtained using this adaptation at the cost of classifier's negative generalisation.

The selection of the LLD features was made using a statistical test that differentiates the promising features on the basis of low significant differences (SD). The Mann-Whitney-U, a non-parametric test, was used to find the SD. Only 38 features were able to satisfy the criteria of *p-value* less than 0.01 and added into the LBOE framework. The evaluation process is carried out on three levels: Word, Syllable, and Phoneme, where similar feature-sets are extracted to compare TTS models. For this, acoustic low-level-descriptors (LLD) are extracted from speech units of each category. The features extracted from each unit belong to four categories:

- **Prosodic-based features:** loudness, pitch (F0), Zero Crossing Rate (ZCR).
- **Spectral-based features:** Psychoacoustic sharpness, Spectral Energy {250-260Hz, 1k-4kHz}, Spectral Roll-Off Points {25%, 50%, 75%, 90%}, Spectral-{entropy, flatness, flux, harmonicity, kurtosis, skewness, variance}.
- **Cepstral-based features:** MFCC-1-16.
- **Voicing-related features:** jitter {local & δ}, shimmer, logHNR, probability of voicing.

There are a total of 38 LLD features (a detailed list of feature-set is given in Appendix E.2). All the LLDs are extracted individually from the isolated frame of speech. These LLDs alone are incapable of capturing any signal dynamics beyond the current frame. The issue was handled by incorporating the derived features and post-processing the features [320]. The feature derivation was done through 2nd order delta regression coefficients. Later, we apply arithmetic mean frame-wise to all LLDs to capture statistical properties that finally produces 76 features for each sound unit.

As the selected features vary in magnitudes, units and range, normalising them to a standard scale is essential for all the models in this study. We apply *z-score* normalisation to normalise features with zero mean and unit variance. The subjective ratings are also scaled with a similar scaling method. During the evaluation, the predicted ratings are scaled back to the origin absolute category rating (ACR).

6.4.3 Automatic Classification

In the third part of LBOE, we compare its performance with other standard parameter sets for classifying the TTS models in order to find the *classification capability* of the proposed feature-set in comparison with the other standard parameter sets. We have used various classifiers to observe the strength of the selected features and confusion of TTS models separately for both datasets. Three classifiers were found to be showing higher accuracies than others: the Support Vector Machine (SVM), Bidirectional-LSTM (Bi-LSTM) and Linear Discriminant Analysis (LDA), compared in Section 6.5.2. Internally, the 10-fold Cross-Validation was incorporated to segregate the speech files of various models into the training and testing data. In order to avoid classifier adaptation, each fold was formed by speech files of different TTS models such that each fold had a uniform distribution.

The classifiers' performances were analysed based on the classification rate, which represents the strength of selected features. We utilise the unweighted average recall (UAR) to achieve the objective [321]. The metric is estimated by taking the mean of both sensitivity and specificity that covers positive as well as negative instances. One more reason to choose UAR as a classification metric is that it has the capability of representing classification accuracy more precisely on unbalanced data as it weights each class equally regardless of its number of samples. The classifier's confusion matrix was used to show how a TTS model is fused with other models and ORIG, which is useful to identify models that are more natural and prosody-rich.

6.4.4 Quality Prediction

The final step of the framework is to perform the task of quality assessment which not only predicts the quality of a synthesised speech but also performs the assessment of various TTS models. First, the instrumental model provides a quality estimate $\hat{\mathcal{Y}}$ based on the physical property attributes $\mathcal{P}_{syn}$ of a synthesised speech. Then, the task of quality assessment is performed using a Cross-Validation setup. The attributes are derived from the (time-variant) LLD features extracted earlier. A one-stage quality prediction model $f()$ is constructed, as shown in Figure 6.6. All the distinct TTS systems were used for the quality-prediction trained on both the datasets. The assessment model is constructed under the Cross-Validation (CV) performance-measuring criteria to observe its generalisation capabilities.

As indicated in Figure 6.6, the property-measurement transforms a synthesised speech signal $\mathcal{S}_{syn}$ into I potential LLD features that are written in matrix notation as $\mathbf{X} = [\mathbf{x}_1^T, ..., \mathbf{x}_n^T, ..., \mathbf{x}_N^T]^T \in \mathbb{R}^{N \times I}$, contains mea-

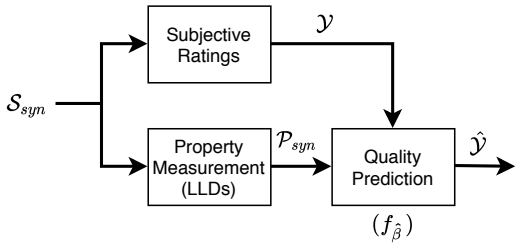


Figure 6.6 Quality-prediction model.

surements from N training signals. Here, each training vector $\mathbf{x}_n^T = \{x_{n,i}\}_{i=1}^I$ consists of I physical-property of n^{th} signal. The corresponding target vector of user-given quality rating is denoted by $\mathbf{y} = [y_1, ..., y_n, ..., y_N]^T$. The quality prediction model is represented as a parametric function $f_{\hat{\boldsymbol{\beta}}}$ which approximates the quality ratings $\hat{y}$ for a speech signal $\mathbf{x}$.

$$\hat{y} = f_{\hat{\boldsymbol{\beta}}}(\mathbf{x}_n) \tag{6.6}$$

Here, the $\hat{\boldsymbol{\beta}}$ is a parameter vector for the assessment model,

$$\hat{\boldsymbol{\beta}} = [\beta_0, \underbrace{\beta_1, ..., \beta_i, ...\beta_I}_{\boldsymbol{\beta}}]^T \tag{6.7}$$

which is composed of regressive intercept β_0 and the weighting vector $\boldsymbol{\beta}$. Consequently, Equation 6.6 can be represented as a weighted superposition of feature-set $\mathbf{x}$ to evaluate it using a multiple linear regression method by:

$$f_{\hat{\boldsymbol{\beta}}}(\mathbf{x}) = \mathbf{x}^T\boldsymbol{\beta} + \beta_0 \tag{6.8}$$

where, $\hat{\boldsymbol{\beta}}$ is determined through least-square. However, multiple linear regression may give instable results if $\mathbf{X}$ has correlated columns affected by overfitting during the training. The assessment models used in the current study have addressed this problem in different ways.

Partial Least Squares Regression (PLS)

PLS regression belongs to the category of a regularised least-squares fit, e.g. principal component regression (PCA) and ridge regression [295]. It performs regression by first finding the less number of orthogonal space-vectors (PLS directions) iteratively by maximising the covariance between different-set of space-vectors and the target vector $\mathbf{y}$. This is equivalent to jointly maximising the variance of each PLS direction $\mathbf{Xr}$ and the squared correlation of the same with $\mathbf{y}$:

$$\underset{r,\|r\|=1}{\arg\max}\ Var(\mathbf{Xr})Corr^2(\mathbf{Xr}, \mathbf{y}). \tag{6.9}$$

We use the PLS regression as a linear prediction model of the form in Equation 6.8. A detailed explanation of PLS regression can be found in [322].

ν-Support Vector Regression (SVR)

ν-SVR, a customised ϵ-SVR, is a regression method based on the Support Vector Machine (SVM) principle [295]. ϵ-SVR uses the ϵ-insensitive error function,

$$|\mathcal{E}|_\epsilon = max\{0, |y - f_{\hat{\boldsymbol{\beta}}}(\mathbf{x}_n) - \epsilon|\} \tag{6.10}$$

which does penalise errors below a chosen priori $\epsilon > 0$. So ν-SVR has the advantage over ϵ-SVR when the approximation's desired accuracy might not be known beforehand, and we want the estimate to be as accurate as possible. The optimised (minimum) size of ϵ is determined via a constant $\nu \in [0, 1]$ which specifies the number of support-vectors used during the regression.

We use the radial-basis-function (RBF) kernel function, $K(\mathbf{x}_m, \mathbf{x}_n) = exp(-\gamma\|\mathbf{x}_n - \mathbf{x}_m\|^2)$, which enhances the prediction accuracy through non-linear representation. Thus, the model performance increases, but at the cost of higher model complexity. We have applied both linear and non-linear Kernel of SVR, denoted as SVR and SVR*, respectively, using python's scikit-learn[16], with default parameter settings.

6.4.5 Assessment of quality prediction model

As the speech synthesised by various TTS models are quite distinct to each other, the assessment of quality prediction model becomes a serious task to look upon. We utilise the Cross-Validation (CV) for the model assessment to monitor over-fitting or under-fitting and provide insight into the model generalisation [295]. Our goal is not only to create a model that fits well on the data (Leave-One-Test-Out CV), but the investigation of how well the trained model generalises to new data (Leave-One-Model-Out CV) is also equally important.

The underlying implementation of Cross-Validation for the model assessment is set up, as shown in Figure 6.7. First, we split the data samples into K groups (folds) under the K-fold CV paradigm. For all possible K combinations, $(K-1)$ groups $\left\{\mathbf{X}^{(k)}_{train}, \mathbf{y}^{(k)}_{train}\right\}$ are considered to *train* the model, the remaining (disjunct) part $\left\{\mathbf{X}^{(k)}_{test}, \mathbf{y}^{(k)}_{test}\right\}$ is used for *testing* the model. Secondly, during the Cross-Validation of each fold, the feature-normalisation details $\boldsymbol{\eta}^{(k)}$ and the parameter vector $\hat{\boldsymbol{\beta}}^{(k)}$ are set and evaluated on the training set and used as it is during the testing. The testing of models incorporates correlation and Root-Mean-Square-Error (RMSE) between $\mathbf{y}^{(k)}_{test}$ and $\hat{\mathbf{y}}^{(k)}_{test}$ as the metric of assessment. The Pearson correlation is calculated between two vectors $\mathbf{y}$ and $\hat{\mathbf{y}}$ as:

[16] sklearn.svm.NuSVR: Uses LIBSVM library at the backend[323].

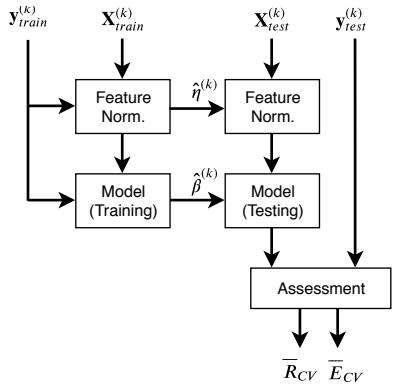


Figure 6.7 CV-setup for the quality assessment model.

$$R(\mathbf{y},\hat{\mathbf{y}}) = \frac{\sum_{n=1}^{N}\left[(y_n - \frac{1}{N}\sum_{n=1}^{N} y_n)(\hat{y}_n - \frac{1}{N}\sum_{n=1}^{N}\hat{y}_n)\right]}{\sqrt{\sum_{n=1}^{N}\left(y_n - \frac{1}{N}\sum_{n=1}^{N} y_n\right)^2 \sum_{n=1}^{N}\left(\hat{y}_n - \frac{1}{N}\sum_{n=1}^{N}\hat{y}_n\right)^2}}. \tag{6.11}$$

The RMSE is estimated by:

$$E(\mathbf{y},\hat{\mathbf{y}}) = \sqrt{\frac{1}{N}\sum_{n=1}^{N}\left(y_n - \hat{y}_n\right)^2} \tag{6.12}$$

The comparison between subjective ratings and their corresponding ratings generated by the model is depicted in Figure 6.7. Here, we have utilised two different CV-setups: (i) Leave-One-Test-Out CV and (ii) Leave-One-Model-Out CV.

Leave-One-Test-Out CV (LOTO)

Under this CV-setup, both evaluation merits: correlation and RMSE, are estimated as arithmetic means of per-fold values:

$$\bar{R}_{LOTO} = \frac{1}{K}\sum_{k=1}^{K} R\big(\mathbf{y},\hat{\mathbf{y}}\big) \tag{6.13}$$

$$\bar{E}_{LOTO} = \frac{1}{K}\sum_{k=1}^{K} E\big(\mathbf{y},\hat{\mathbf{y}}\big) \tag{6.14}$$

We have partitioned the data into three parts (K=3) in LOTO-CV for analysing the models.

Leave-One-Model-Out CV (LOMO)

In this CV-setup, the partition is carried out based on the type of TTS-model. In each test, samples belonging to a specific TTS-model are kept separately, not to be used during training a model. The correlation and RMSE are estimated for each TTS model separately as below:

$$\bar{R}_{LOMO}(m) = R\big(\mathbf{y}_m, \hat{\mathbf{y}}_m\big), \tag{6.15}$$

$$\bar{E}_{LOMO}(m) = E\big(\mathbf{y}_m, \hat{\mathbf{y}}_m\big). \tag{6.16}$$

where m denotes the TTS model, which is excluded during the training under LOMO CV-setup.

6.5 Results & Analysis

Table 6.2 Intelligibility Test: Pair-wise Comparison of all TTS models.

TTS Models	CMU-HMM	CMU-CLU	CMU-DNN	CMU-USS	IITM-HMM	IITM-CLU	IITM-DNN	IITM-USS	HUMAN
CMU-HMM	0.00(1.00)	11.04(0.00)	12.69(0.00)	13.67(0.00)	0.13(0.72)	15.33(0.00)	16.19(0.00)	27.50(0.00)	21.43(0.00)
CMU-CLU	11.04(0.00)	0.00(1.00)	0.03(0.86)	0.17(0.68)	13.84(0.00)	0.30(0.58)	0.41(0.52)	3.85(0.05)	1.82(0.18)
CMU-DNN	12.69(0.00)	0.03(0.86)	0.00(1.00)	0.06(0.81)	15.77(0.00)	0.15(0.70)	0.22(0.64)	3.35(0.07)	1.44(0.23)
CMU-USS	13.67(0.00)	0.17(0.68)	0.06(0.81)	0.00(1.00)	16.78(0.00)	0.02(0.90)	0.05(0.83)	2.36(0.12)	0.86(0.35)
IITM-HMM	0.13(0.72)	13.84(0.00)	15.77(0.00)	16.78(0.00)	0.00(1.00)	18.71(0.00)	19.70(0.00)	32.06(0.00)	25.40(0.00)
IITM-CLU	15.33(0.00)	0.30(0.58)	0.15(0.70)	0.02(0.90)	18.71(0.00)	0.00(1.00)	0.01(0.93)	2.10(0.15)	0.68(0.41)
IITM-DNN	16.19(0.00)	0.41(0.52)	0.22(0.64)	0.05(0.83)	19.70(0.00)	0.01(0.93)	0.00(1.00)	1.89(0.17)	0.56(0.46)
IITM-USS	27.50(0.00)	3.85(0.05)	3.35(0.07)	2.36(0.12)	32.06(0.00)	2.10(0.15)	1.89(0.17)	0.00(1.00)	0.37(0.54)
HUMAN	21.43(0.00)	1.82(0.18)	1.44(0.23)	0.86(0.35)	25.40(0.00)	0.68(0.41)	0.56(0.46)	0.37(0.54)	0.00(1.00)

† Values in the table $p(q)$: p is F-value and q is p-value obtained from the ANOVA test.

Table 6.3 Subjective Evaluation: Question-wise Mean & Standard-Deviation of Various Models.

TTS Models	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10
CMU-HMM	2.43(0.55)	2.46(0.57)	2.35(0.52)	2.26(0.47)	2.11(0.41)	2.19(0.37)	2.23(0.43)	2.18(0.39)	2.21(0.39)	2.21(0.42)
CMU-CLU	4.01(0.51)	3.95(0.53)	3.90(0.52)	3.73(0.48)	3.60(0.44)	3.66(0.42)	3.73(0.42)	3.68(0.44)	3.72(0.44)	3.69(0.43)
CMU-DNN	4.16(0.52)	4.22(0.48)	4.05(0.56)	3.91(0.42)	3.83(0.36)	3.52(0.52)	3.69(0.40)	3.88(0.45)	3.73(0.38)	3.77(0.43)
CMU-USS	4.44(0.51)	4.34(0.53)	4.23(0.53)	4.10(0.49)	3.95(0.48)	4.02(0.48)	4.11(0.45)	4.04(0.49)	4.12(0.46)	4.13(0.49)
IITM-HMM	3.15(0.57)	3.14(0.58)	3.09(0.57)	2.96(0.50)	2.93(0.47)	2.94(0.44)	3.01(0.45)	3.00(0.47)	3.03(0.47)	2.97(0.49)
IITM-CLU	5.23(0.42)	5.25(0.41)	5.19(0.43)	5.04(0.42)	4.89(0.41)	4.87(0.40)	4.84(0.39)	5.03(0.39)	4.96(0.37)	4.98(0.41)
IITM-DNN	5.47(0.45)	5.69(0.37)	5.23(0.51)	5.40(0.33)	5.54(0.43)	5.36(0.48)	5.27(0.41)	5.67(0.35)	5.18(0.42)	5.27(0.38)
IITM-USS	6.05(0.30)	6.06(0.29)	6.05(0.29)	5.96(0.30)	5.78(0.34)	5.84(0.33)	5.82(0.34)	5.92(0.30)	5.86(0.34)	5.88(0.31)
HUMAN	6.58(0.28)	6.60(0.28)	6.61(0.26)	6.52(0.27)	6.37(0.31)	6.41(0.34)	6.33(0.38)	6.45(0.33)	6.39(0.35)	6.45(0.32)

† Values in the table $p(q)$: p is mean and q is standard deviation.

Table 6.4 Subjective Evaluation: Mean & Standard Deviation for Comprehension, Naturalness and Prosody.

TTS Models	Comprehensibility [Q1-Q4]	Naturalness [Q5-Q8]	Prosody [Q9-Q10]
CMU-HMM	2.38(0.53)	2.18(0.40)	2.21(0.40)
CMU-CLU	3.90(0.52)	3.67(0.43)	3.71(0.44)
CMU-DNN	4.07(0.58)	3.83(0.51)	3.95(0.56)
CMU-USS	4.28(0.53)	4.03(0.48)	4.12(0.48)
IITM-HMM	3.08(0.56)	2.97(0.46)	3.00(0.48)
IITM-CLU	5.18(0.43)	4.91(0.40)	4.97(0.39)
IITM-DNN	5.88(0.40)	5.62(0.38)	5.17(0.35)
IITM-USS	6.03(0.30)	5.84(0.33)	5.87(0.33)
HUMAN	6.58(0.28)	6.39(0.34)	6.42(0.33)

† Values in the table $p(q)$: p is mean and q is standard deviation.

6.5.1 Subjective & Objective Evaluation Results

After obtaining responses from the participants in the form of ratings as well as written text, we compare all the TTS models to each other on various statistical measures. First, a pairwise comparison of all the TTS models is made through the ANOVA test on their estimated word error rates (WER). Later, a categorised comparison of all the models is shown on comprehensibility, naturalness and prosody based on the listener's rating.

The subjective similarity of two speech synthesisers is achieved on the basis of significant-difference of WER estimated between corresponding TTS models using the one-way ANOVA test. Table 6.2 shows the similarity among all possible pairs of TTS models as well as Human speech. It also reflects that human speech has a significant difference with other models in decreasing order for HMM, CLU, DNN, and USS on both the datasets. In pairwise comparison through ANOVA test, we found a significant difference of CMU-HMM with CMU-CLU ($F(1, 460) = 11.04, p < 0.0001$), CMU-DNN ($F(1, 460) = 12.69, p < 0.0001$), CMU-USS ($F(1, 460) = 13.67, p < 0.0001$) as well as HUMAN ($F(1, 460) = 21.43, p < 0.0001$) which shows that CMU-HMM model is far from the human and other TTS models in terms of WER score. For CMU-CLU, the distance with CMU-DNN ($F(1, 460) = 0.03, p = 0.86$), CMU-USS ($F(1, 460) = 0.17, p = 0.68$)) and HUMAN ($F(1, 460) = 1.82, p = 0.18$) is lower which represents CMU-CLU voice is closer to CMU-DNN but not to human and CMU-USS models. Similarly for CMU-DNN, the distance with CMU-USS ($F(1, 460) = 0.06, p = 0.81$) and HUMAN ($F(1, 460) = 1.44, p = 0.23$) is also lower compared to human speech. CMU-USS is found to be closer to HUMAN with a distance of ($F(1, 460) = 0.86, p = 0.35$) than other models. Similar proximity behaviour is observed when CMU models are compared with the IITM models.

On the IITM based models, the ANOVA test follows similar pattern and shows significant distance of IITM-HMM with IITM-CLU, ($F(1, 460) = 18.71, (p < 0.0001)$), IITM-DNN ($F(1, 460) = 19.70, p < 0.0001$), IITM-USS ($F(1, 460) = 32.06, p < 0.0001$) and HUMAN ($F(1, 460) = 25.40, p < 0.0001$). Next IITM-CLU is comparatively lower distant with IITM-USS ($F(1, 460) = 2.10, p = 0.15$), IITM-DNN ($F(1, 460) = 0.01, p = 0.93$) and HUMAN ($F(1, 460) = 0.68, p = 0.41$). Similar to IITM-CLU, IITM-DNN is also significantly close to IITM-USS ($F(1, 460) = 1.89, p = 0.17$) and HUMAN($F(1, 460) = 0.56, p = 0.46$). Here again, IITM-USS, is closest to HUMAN speech with $F(1, 460) = 0.37(p = 0.54)$ measure of distance. Overall, the intelligibility test of listener's response indicates that USS models are more natural and human-like than the other TTS models. CLU and DNN are found to be quite similar to each other. HMM model's WER score is worst among all the models.

Verifying, mean (M) and standard deviation (SD) of the WER, we observe that the human speech has significantly lower WER (M=20.63, SD=11.64) than CMU-HMM (M=32.02, SD=13.30), IITM-HMM (M=32.47, SD=13.17), CMU-CLU (M=28.12, SD=11.93), IITM-CLU (M=27.94, SD=11.41) and CMU-DNN (M=28.02,

SD=11.39) IITM-DNN (M=27.32, SD=12.37), CMU-USS(M=26.65, SD=12.02), IITM-USS(M=25.98, SD=11.35) based models.

In the second part of subjective evaluation, the models are compared on the basis of rating obtained on the MOS-X questionnaire, as shown in Table 6.3. It is evident from Table 6.3-6.4 that the parametric speech synthesiser's speech quality is still far lower compared to concatenative synthesisers. Analysing Q1-Q10, under the 7-point scale, Human speech is rated in range (6.3-6.6). For IITM dataset, HMM, CLU, DNN and USS based models are rated in the range (2.9-3.2), (4.8-5.3), (5.2-5.7) and (5.8-6.1), respectively. Similarly, CMU based HMM, CLU, DNN and USS models have obtained ratings in range (2.1-2.5), (3.6-4.0), (3.5-4.2) and (4.1-4.5), respectively.

Table 6.4 presents the consolidated results under MOS-X categories of comprehensibility, naturalness and prosody. For both datasets, listeners rated the USS model higher. Although the models trained through CLU and DNN are not very natural and prosodic-rich, they are quite close to USS as per the rating. HMM-based models show significantly lower performance than the other models. Based on the subjective evaluation, the USS model trained on both datasets seems to be performed well than the other TTS models in terms of comprehensibility, naturalness and prosody. Now, we aim to achieve the same outcome through a set of objective evaluations.

Table 6.5 Objective Evaluation: Models Trained on CMU.

Evaluation Metrics	**CMU-HMM**	**CMU-CLU**	**CMU-DNN**	**CMU-USS**
MCD	7.6675(0.3093)	7.1082(0.3158)	5.1585(0.2179)	2.0432(1.0201)
F0-RMSE	81.6182(48.6134)	44.6793(27.3087)	35.2275(22.1971)	19.1844(17.9790)
SNR	0.5740(0.0981)	1.8000(0.0804)	1.3537(0.1199)	1.3352(0.0426)
Global SNR	1.8261(0.0657)	0.4122(0.1267)	1.1545(0.1353)	1.1200(0.0533)
Segmental SNR	0.4329(0.1139)	0.7268(0.1122)	0.5244(0.1429)	1.2873(0.1205)
frequency-weighted segmental SNR	1.3392(0.0961)	0.4117(0.1217)	1.3283(0.1217)	1.3903(0.0859)
Broadband SNR	0.5794(0.0999)	1.6757(0.0787)	1.2291(0.1138)	1.0854(0.0214)
LPCC-Spectral Distortion	0.2243(0.1177)	1.2792(0.1149)	0.8888(0.1503)	0.3753(0.1109)
Log-Spectral Distortion	1.5140(0.1137)	1.3631(0.1377)	0.5556(0.1930)	0.9462(0.1062)
Itakura-Saito Spectral Distortion	1.0268(0.0265)	0.7306(0.0995)	0.0830(0.0737)	0.1542(0.0921)
WSS	1.5272(0.1590)	1.3212(0.1408)	1.0910(0.1240)	0.4949(0.1199)
lagDiff	1.3780(0.1170)	0.9118(0.0639)	0.4412(0.0205)	0.3700(0.0551)
Bark-Spectral Distortion	1.0552(0.0409)	0.0458(0.0192)	0.1327(0.0583)	0.3006(0.0652)
PESQ	0.1699(0.6065)	0.0343(0.2588)	0.2778(0.4930)	0.1113(0.3095)

† Values in the table $p(q)$: p is mean and q is standard deviation.

Table 6.5-6.6 presents the objective evaluation results for both CMU and IITM based TTS models. Except for Mel-Cepstral Distortion (MCD) and root mean square error of F0 (F0-RMSE), none of the objective evaluation measures is able to distinguish the TTS models. Based on the MCD and F0-RMSE scores, it is evident from both the table that USS delivers the best result. They also support the subjective evaluation outcome that DNN based TTS models are better than CLU based models. In comparison to all the TTS models, HMM-based models are not up to the mark.

Table 6.6 Objective Evaluation: Models Trained on IITM.

Evaluation Metrics	IITM-HMM	IITM-CLU	IITM-DNN	IITM-USS
MCD	7.7559(0.7662)	6.9592(0.3612)	4.9753(0.3840)	3.5474(0.8665)
F0-RMSE	36.8680(34.3779)	34.1950(28.9927)	27.7815(21.4731)	18.2559(25.6142)
SNR	0.6851(0.1305)	1.5956(0.1178)	0.9743(0.1256)	0.2796(0.0479)
Global SNR	1.6001(0.0281)	0.5667(0.1132)	1.2322(0.1246)	1.2487(0.0455)
Segmental SNR	0.6692(0.1138)	0.5709(0.1431)	0.5081(0.1372)	1.2044(0.1402)
frequency-weighted segmental SNR	0.7250(0.0968)	0.9177(0.1390)	1.3179(0.1186)	1.5437(0.0889)
Broadband SNR	0.5728(0.1364)	1.0646(0.1223)	0.6993(0.1235)	1.0347(0.0240)
LPCC-Spectral Distortion	1.1511(0.0832)	0.7018(0.0854)	0.7690(0.1176)	0.1622(0.0787)
Log-Spectral Distortion	1.3071(0.0979)	1.0164(0.1153)	0.8320(0.1033)	0.6416(0.1748)
Itakura-Saito Spectral Distortion	1.0000(0.0000)	0.6343(0.0000)	0.0185(0.0000)	0.0210(0.0340)
WSS	1.2577(0.0373)	1.0343(0.1182)	1.1142(0.1277)	0.5247(0.1574)
lagDiff	1.3717(0.0492)	0.5975(0.0594)	0.6510(0.0383)	0.8955(0.0391)
Bark-Spectral Distortion	1.1995(0.1016)	0.5093(0.0959)	0.2549(0.0557)	0.4569(0.1689)
PESQ	0.4315(0.7099)	0.3398(0.3564)	0.6810(0.2697)	2.1024(0.9549)

† Values in the table $p(q)$: p is mean and q is standard deviation.

The rest of the objective evaluation metrics are taken from various distortion measures used in the Speech-Coding [313]. As expected, the speech coding metrics do not clearly distinguish the quality of the models used in the area of speech synthesis. Their performance is inconsistent on both datasets. For example, BSD show CMU-CLU best for CMU dataset and IITM-DNN for IITM dataset. Similarly, PESQ, LPCC-Spectral Distortion, Itakura-Saito Spectral Distortion, lagDiff and Log-Spectral Distortion and other SNR-related distortion measures are also unable to distinguish the models in an expected order in terms of performance and quality.

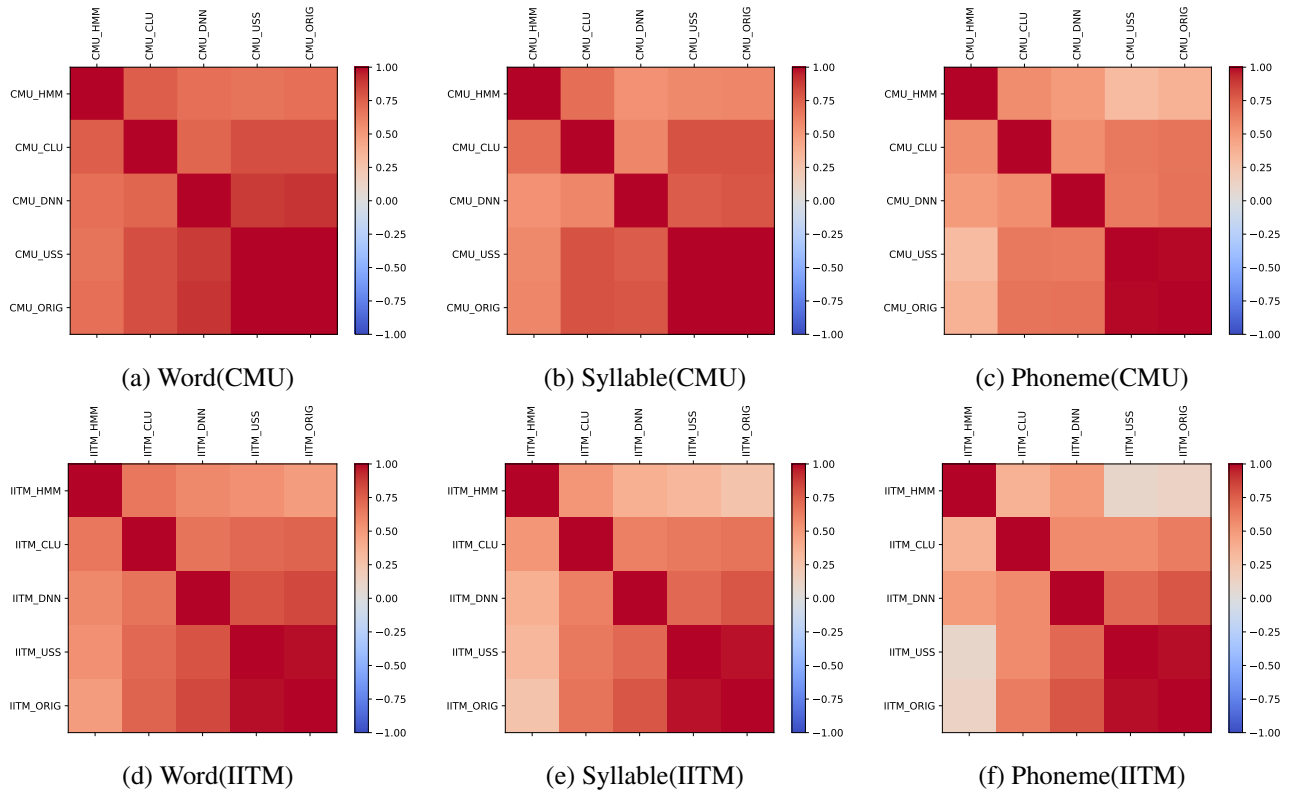
6.5.2 Learning-Based Objective Evaluation

After discussing the subjective and objective evaluation results, this subsection demonstrates the results of the proposed LBOE framework and its usefulness. The proposed Learning-Based Objective Evaluation starts with analysing various TTS models on the proposed feature-set in comparison with the standard parameters set (see Table 6.7). The openSMILE toolkit facilitates the extractions of such parameter-set from the speech.

Table 6.7 Comparison of various standard parameter-set based on the number of features.

Parameter-Set	#Features
InterSpeech09 Emotion Challenge [324]	384
InterSpeech10 Paralinguistics Challenge [325]	1,582
InterSpeech11 Speaker State Challenge [326]	4,368
InterSpeech12 Speaker Trait Challenge [327]	6,125
InterSpeech (13&16) Computational Paralinguistics Challenge [328, 321]	6,373
AVEC13 [329]	2,268
GeMAPS [330]	62
eGeMAPS [330]	88
Proposed feature-set	**76**

In the Learning-Based Objective Evaluation, first of all, we extract the proposed LLD feature-set, covering voicing, energy and spectral properties from the speech files of all the TTS models. Table E.1 and E.2 represent



(a) Word(CMU) (b) Syllable(CMU) (c) Phoneme(CMU)

(d) Word(IITM) (e) Syllable(IITM) (f) Phoneme(IITM)

Figure 6.8 Mean-correlation of various TTS models shown for CMU & IITM separately.

sample mean & standard deviation of word-level features extracted from ORIG, HMM, CLU, DNN, and USS models of TTS trained on both CMU and IITM in Appendix E.3. Based on the raw mean values of the proposed LLD features-set, it is evident that USS based models are closer to ORIG (human) speech on both the datasets. In order to observe the statistical proximity of the models, we have also drawn a correlation graph separately at Phoneme, Syllable and Word level features for CMU (see Figure 6.8a–6.8c) and IITM (see Figure 6.8d–6.8f). Dataset-wise, both the triplets signify the decreasing power of correlation in the order of phoneme, syllable and word-level features. One major observation present in all the sub-figures of Figure 6.8 is that whatever the dataset is, there is a decreasing order in the correlation of HMM, CLU, DNN and USS with ORIG. Based on the proposed feature-set, USS synthesiser output is highly correlated with the human voice. On the contrary, HMM is least likely to be similar to the human voice. On the other hand, DNN output seems to be a bit better than CLU.

In order to observe the strength of the proposed feature-set, we have performed the classification task in comparison with the other standard parameter sets. Table 6.8 and 6.9 summarise the classification results on the features extracted at all three levels: Phoneme, Syllable and Word. The classification is performed separately on the TTS models belonging to CMU and IITM families. It is observed that at the word, syllable, and phoneme levels, our proposed feature-set has consistently performed better or equal compared to the other parameter sets. Support Vector Machine (SVM) with RBF-kernel, Linear Discriminant Analysis (LDA) and Bidirectional-LSTM (Bi-LSTM) were the top-performing classification methods.

Table 6.8 Classification Accuracies (%) of CMU Models On (Word,Syllable,Phoneme)-Level Features.

Feature-Sets	Word			Syllable			Phoneme		
	LDA	SVM	Bi-LSTM	LDA	SVM	Bi-LSTM	LDA	SVM	Bi-LSTM
GeMAPS	65.69	68.75	67.28	59.15	65.31	59.46	49.87	58.45	48.64
eGeMAPS	70.13	72.82	72.63	65.81	69.77	68.47	57.96	68.09	61.65
IS09Emo	75.14	73.22	75.04	72.33	73.47	69.31	63.38	73.18	60.56
IS10Paral	73.42	72.18	69.56	69.52	69.48	66.37	60.22	64.46	56.99
AVEC13	75.27	72.41	73.97	75.43	73.29	72.76	**72.75**	72.54	**69.46**
IS11SS	74.71	73.53	75.85	74.65	72.9	**73.52**	72.63	72.09	67.73
IS12ST	72.75	72.04	**75.95**	**75.84**	**74.33**	62.43	72.57	71.71	66.65
IS13ComParE	74.18	73.04	74.78	73.47	72.76	71.61	71.87	70.66	68.79
Proposed feature-set	**75.53**	**74.45**	75.25	71.77	73.77	72.35	61.36	**74.07**	59.22

Table 6.9 Classification Accuracies (%) of IITM Models On (Word,Syllable,Phoneme)-Level Features.

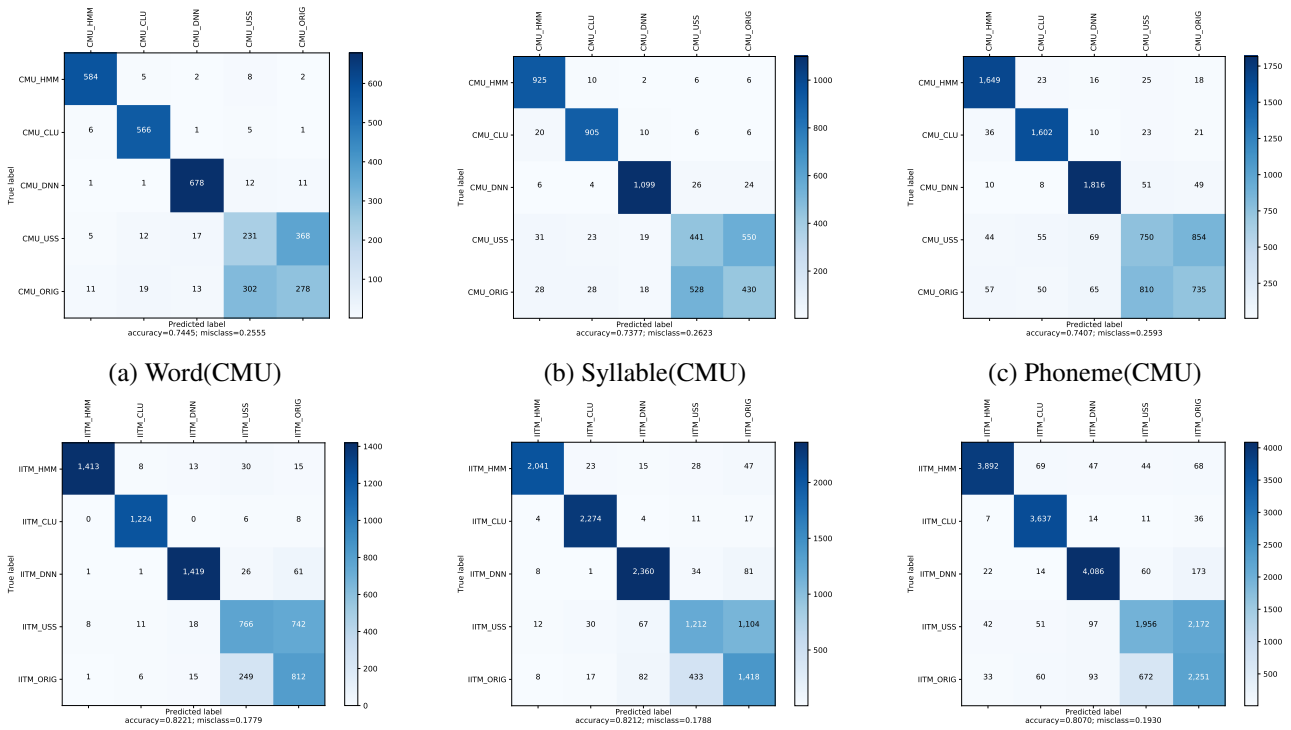
Feature-Sets	Word			Syllable			Phoneme		
	LDA	SVM	Bi-LSTM	LDA	SVM	Bi-LSTM	LDA	SVM	Bi-LSTM
GeMAPS	71.43	73.67	71.47	67.37	72.52	67.2	59.11	67.25	58.53
eGeMAPS	75.56	76.75	73.81	68.39	74.62	70.17	62.79	73.23	65.32
IS09Emo	82.85	**82.91**	77.79	77.59	81.32	72.17	66.44	79.84	64.64
IS10Paral	81.88	79.58	76.05	77.34	78.18	72.29	66.13	72.13	65.15
AVEC13	**86.77**	80.55	54.48	**85.30**	81.76	**77.54**	**79.04**	79.09	71.28
IS11SS	82.89	77.19	74.73	80.74	76.98	74.01	75.93	76.08	66.56
IS12ST	83.43	77.02	75.19	82.63	77.26	73.38	76.72	76.11	**71.49**
IS13ComParE	82.43	76.12	74.49	82.43	78.09	73.49	76.22	76.05	70.85
Proposed feature-set	80.45	82.21	**83.20**	76.90	**82.12**	73.81	67.21	**80.70**	61.53

Table 6.10 Classification Timing (Minutes) of CMU Models On (Word,Syllable,Phoneme)-Level Features.

Feature-Sets	Word			Syllable			Phoneme		
	LDA	SVM	Bi-LSTM	LDA	SVM	Bi-LSTM	LDA	SVM	Bi-LSTM
GeMAPS	**0.01**	**0.24**	**0.23**	**0.01**	**0.84**	**0.36**	**0.02**	3.19	**0.62**
eGeMAPS	0.02	0.31	0.24	0.02	1.06	0.38	0.02	3.65	0.66
IS09Emo	0.03	1.4	0.31	0.05	3.88	0.48	0.08	16.74	0.95
IS10Paral	0.45	5.68	0.78	0.61	15.39	1.25	0.8	38.87	1.76
AVEC13	1.11	8.15	1.09	1.41	22.36	1.64	1.9	8.33	2.69
IS11SS	5.78	16.15	1.88	6.5	43.05	2.74	7.46	42.88	4.06
IS12ST	11.19	22.07	2.26	12.76	57.55	3.5	14.63	50.38	6.05
IS13ComParE	14.31	24.1	2.49	16.67	4.35	3.93	19.19	13.06	6.94
Proposed feature-set	**0.01**	0.28	**0.23**	0.02	0.9	0.37	**0.02**	**2.94**	0.65

As SVM with RBF-kernel outperforms the other classifiers, we also investigate its confusion matrix (see Figure 6.9). In all possible scenarios, USS and ORIG are completely fused, which entails that USS's synthesised speech is highly natural and prosodically rich as a human voice. Thus, the classification accuracy supports our claim that the proposed minimal set of features can distinguish speech files belonging to various TTS models.

Accuracy alone does not fulfil the purpose; the evaluation framework should also be time-efficient. So, we also compare the performance of the feature-sets on the basis of time taken in classification. The time taken



(a) Word(CMU) (b) Syllable(CMU) (c) Phoneme(CMU)

(d) Word(IITM) (e) Syllable(IITM) (f) Phoneme(IITM)

Figure 6.9 Confusion Matrices of SVM on Classifying TTS Models performed separately for CMU & IITM datasets.

Table 6.11 Classification Timing (Minutes) of IITM Models On (Word,Syllable,Phoneme)-Level Features.

Feature-Sets	**Word**			**Syllable**			**Phoneme**		
	LDA	**SVM**	**Bi-LSTM**	**LDA**	**SVM**	**Bi-LSTM**	**LDA**	**SVM**	**Bi-LSTM**
GeMAPS	0.03	**1.15**	0.5	**0.02**	**4.12**	**0.74**	**0.03**	13.45	**1.26**
eGeMAPS	**0.02**	1.49	0.51	0.03	4.7	**0.74**	0.04	15.44	1.31
IS09Emo	0.06	6.62	0.67	0.09	19.2	1.08	0.17	39.35	2.32
IS10Paral	0.83	24.55	1.79	1.13	10.07	2.85	1.44	49.32	4.3
AVEC13	1.9	36.47	2.53	2.46	40.68	4.13	3.38	46.21	6.88
IS11SS	8.92	10.8	4.75	10.72	14.53	7.82	12.53	39.31	11.54
IS12ST	18.62	36.51	6.73	21.1	28.6	11.05	20.56	25.44	14.79
IS13ComParE	19.39	27.03	5.55	22.09	13.54	9.21	26.11	15.39	14.22
Proposed feature-set	**0.02**	1.49	**0.49**	0.03	4.52	0.75	0.04	**13.42**	1.3

in training and testing for each feature-set is summed up for their comparative analysis. The classification task on each feature-set is executed on a machine with an *Octa-core Intel Xeon E5-2630* processor and *256 GB RAM*. The execution time of all the feature-set is shown in Table 6.10 and 6.11 separately for CMU and IITM based TTS models on the features extracted at the word, syllable and phoneme levels. From both the tables, GeMAPS and the *proposed feature-set* are found to be the most time-efficient feature-sets. If we compare the classification accuracy of both, the proposed feature-set is far better than the GeMAPS. Hence, the proposed feature-set not only delivers comparable classification accuracy but is also time-efficient, which is highly desired for an evaluation framework.

Table 6.12 Evaluation matrics of quality prediction models for the **LOTO-CV** test-cases on (Word,Syllable,Phoneme)-Level Features of **CMU** models.

Evaluation Metrics	Model	Comprehensibility			Naturalness			Prosody		
		Word	**Syl**	**Pho**	**Word**	**Syl**	**Pho**	**Word**	**Syl**	**Pho**
$\bar{R}_{LOTO}$	PLS	0.59	0.52	0.44	0.59	0.51	0.44	0.58	0.51	0.44
	SVR	0.78	0.73	0.60	0.79	0.73	0.61	0.79	0.73	0.61
	SVR*	**0.86**	**0.83**	**0.81**	**0.87**	**0.84**	**0.82**	**0.83**	**0.81**	**0.79**
$\bar{E}_{LOTO}$	PLS	1.08	1.14	1.20	1.09	1.15	1.20	1.10	1.17	1.22
	SVR	0.84	0.92	1.07	0.83	0.92	1.08	0.85	0.94	1.10
	SVR*	**0.68**	**0.71**	**0.74**	**0.67**	**0.70**	**0.73**	**0.69**	**0.72**	**0.75**

Table 6.13 Evaluation matrics of quality prediction models for the **LOTO-CV** test-cases on (Word,Syllable,Phoneme)-Level Features of **IITM** models.

Evaluation Metrics	Model	Comprehensibility			Naturalness			Prosody		
		Word	**Syl**	**Pho**	**Word**	**Syl**	**Pho**	**Word**	**Syl**	**Pho**
$\bar{R}_{LOTO}$	PLS	0.71	0.63	0.48	0.70	0.62	0.48	0.70	0.62	0.47
	SVR	0.85	0.81	0.71	0.84	0.80	0.70	0.84	0.80	0.70
	SVR*	**0.90**	**0.87**	**0.83**	**0.90**	**0.85**	**0.83**	**0.87**	**0.86**	**0.84**
$\bar{E}_{LOTO}$	PLS	0.94	1.02	1.17	0.92	1.01	1.14	0.93	1.02	1.15
	SVR	0.70	0.78	0.94	0.69	0.77	0.92	0.71	0.78	0.93
	SVR*	**0.59**	**0.60**	**0.62**	**0.57**	**0.58**	**0.61**	**0.58**	**0.59**	**0.62**

Results of the assessment of quality prediction models

After showing the strength of the proposed feature-set, we analyse the performance of the quality prediction models through Cross-Validation setups: LOTO-CV and LOMO-CV. The results of LOTO-CV are discussed in Table 6.12-6.13, and LOMO-CV in Table 6.14-6.15. The assessment of the correlation coefficient is done through a two-tailed *t-test* ($H_{NULL} : R = 0$, $H_1 : R \neq 0$) [331]. For LOTO-CV, a sample size of $N = 3070$ signals (word-level), a correlation larger than .62 is considered significant at 95% confidence interval ($p < 0.05$). For $N = 5900$ signals (syllable-level), a correlation larger than .56 is considered significant. For $N = 11580$ signals (phoneme-level), a correlation larger than .48 is considered significant. On the other hand, for LOMO-CV, a correlation higher than 0.63 is considered significant. All correlation values given in Table (6.12 to 6.15) are the *averaged* CV correlations.

Both LOTO-CV and LOMO-CV have been considered to observe the strength of model assessment towards the prediction of various subjective-evaluation dimensions, e.g. comprehensibility, naturalness and prosody (see Table 6.12-6.15). For each dimension, the correlation declines along with the Word, Syllable and Phoneme based features for all the model types. This is because the realisation of phoneme or syllable is strongly influ-

enced by its adjacent units [332]. Here, ratings for all the dimensions are kept on the same relative scale, which helps retain the bias and variance proportions.

Table 6.14 Evaluation matrics of quality prediction model (**SVR***) for the **LOMO-CV** test-cases on (Word,Syllable,Phoneme)-Level Features of **CMU** models.

Evaluation	**CMU**	**Comprehensibility**			**Naturalness**			**Prosody**		
Metrics	**TTS-Models**	**Word**	**Syl**	**Pho**	**Word**	**Syl**	**Pho**	**Word**	**Syl**	**Pho**
$\bar{R}_{LOMO}$	HMM	0.41	0.33	0.30	0.37	0.33	0.27	0.31	0.29	0.26
	CLU	0.75	0.73	0.71	0.76	0.74	0.72	0.66	0.61	0.58
	DNN	0.73	0.71	0.69	0.60	0.58	0.53	0.59	0.51	0.42
	USS	0.79	0.71	0.76	0.74	0.69	0.65	0.65	0.63	0.59
	ORIG	0.42	0.28	0.43	0.21	0.34	0.32	0.16	0.28	0.18
$\bar{E}_{LOMO}$	HMM	0.75	0.89	0.93	0.87	0.92	1.12	0.98	1.22	1.46
	CLU	0.29	0.33	0.39	0.31	0.36	0.39	0.35	0.37	0.43
	DNN	0.27	0.31	0.38	0.29	0.31	0.38	0.38	0.42	0.47
	USS	0.32	0.31	0.35	0.32	0.37	0.34	0.36	0.42	0.51
	ORIG	0.91	0.85	1.08	0.93	0.77	0.84	1.18	1.20	1.60

† Underlined values are the results of TTS models with inner subjective ratings.

Table 6.15 Evaluation matrics of quality prediction models (**SVR***) for the **LOMO-CV** test-cases on (Word,Syllable,Phoneme)-Level Features of **IITM** models.

Evaluation	**IITM**	**Comprehensibility**			**Naturalness**			**Prosody**		
Metrics	**TTS-Models**	**Word**	**Syl**	**Pho**	**Word**	**Syl**	**Pho**	**Word**	**Syl**	**Pho**
$\bar{R}_{LOMO}$	HMM	0.42	0.40	0.31	0.40	0.41	0.36	0.47	0.44	0.41
	CLU	0.71	0.66	0.67	0.79	0.76	0.74	0.67	0.63	0.59
	DNN	0.61	0.58	0.55	0.55	0.54	0.49	0.53	0.50	0.46
	USS	0.62	0.63	0.63	0.69	0.66	0.64	0.68	0.64	0.66
	ORIG	0.37	0.42	0.40	0.44	0.42	0.37	0.44	0.39	0.41
$\bar{E}_{LOMO}$	HMM	0.69	0.74	0.73	0.85	0.70	0.92	0.87	0.95	0.83
	CLU	0.25	0.32	0.37	0.33	0.34	0.38	0.35	0.37	0.43
	DNN	0.26	0.29	0.33	0.26	0.34	0.37	0.27	0.29	0.35
	USS	0.21	0.25	0.32	0.24	0.28	0.31	0.25	0.31	0.33
	ORIG	0.78	0.83	0.89	0.73	0.95	0.91	0.76	0.95	1.04

† Underlined values are the results of TTS models with inner subjective ratings.

Word-based features often show lower errors than others, which denotes its superiority in predicting the ratings. However, Phoneme-based features were found to be the lowest among the dimensions. The trend is followed independently over all the regression models. The scatterplots in Figure 6.10 and 6.11 demonstrate the differences graphically. Normalising the error to the observed range can be used to compensate for this effect.

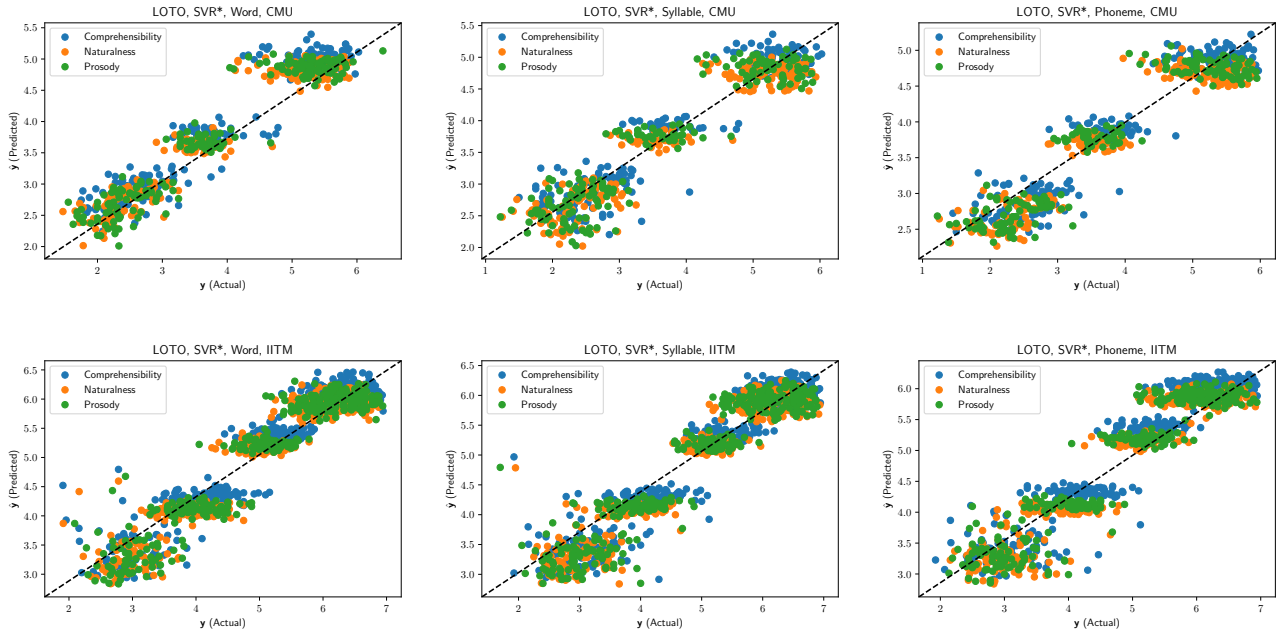


Figure 6.10 Scatterplots for LOTO-CV performed separately for CMU & IITM shown in two rows correspond to three feature-types Word, Syllabe and Phoneme respectively.

Comparison on assessment model-type

In both LOTO-CV and LOMO-CV, model assessment by SVR with RBF-kernel (SVR*) is represented as the winning model both in terms of correlation and error, with few exceptions. Based on the two-tailed $hypothesis$-test, the results denote superiority of the non-linear model SVR* compared to the linear models: SVR and PLS. The graphical comparison of the assessment models is presented in Figure 6.10 and 6.11.

Leave-One-Model-Out CV (LOMO-CV) Results

In order to show the generalisation capability of the LBOE towards unknown TTS models, we carried out the model assessment through LOMO-CV setup. The assessment results obtained via LOMO-CV are listed in Table 6.14 and 6.15. As the performance of various assessment models shows SVR* is best in characterising the aspects of various TTS model types, we limit our analysis only to SVR*. Comparing the performance of SVR* among five TTS-models on both the datasets, substantial similarity can be observed in different subjective-properties, which indicates their structural similarity under features at various levels, e.g. word, syllable or phoneme-based. One TTS-model is kept out during each test, the per-test performance of LOMO is notably consistent than in the LOTO case.

It is observed that LOMO-CV delivers higher performance for inner TTS-models, e.g. CMU-CLU, CMU-DNN, CMU-USS or IITM-CLU, IITM-DNN, IITM-USS in terms of relative rating and worst for other models in outer-range of the ACR, i.e. HMM and ORIG. These results show that an assessment model can better generalise an unknown TTS-model if a large number of samples from different TTS-models are used in training. Figure

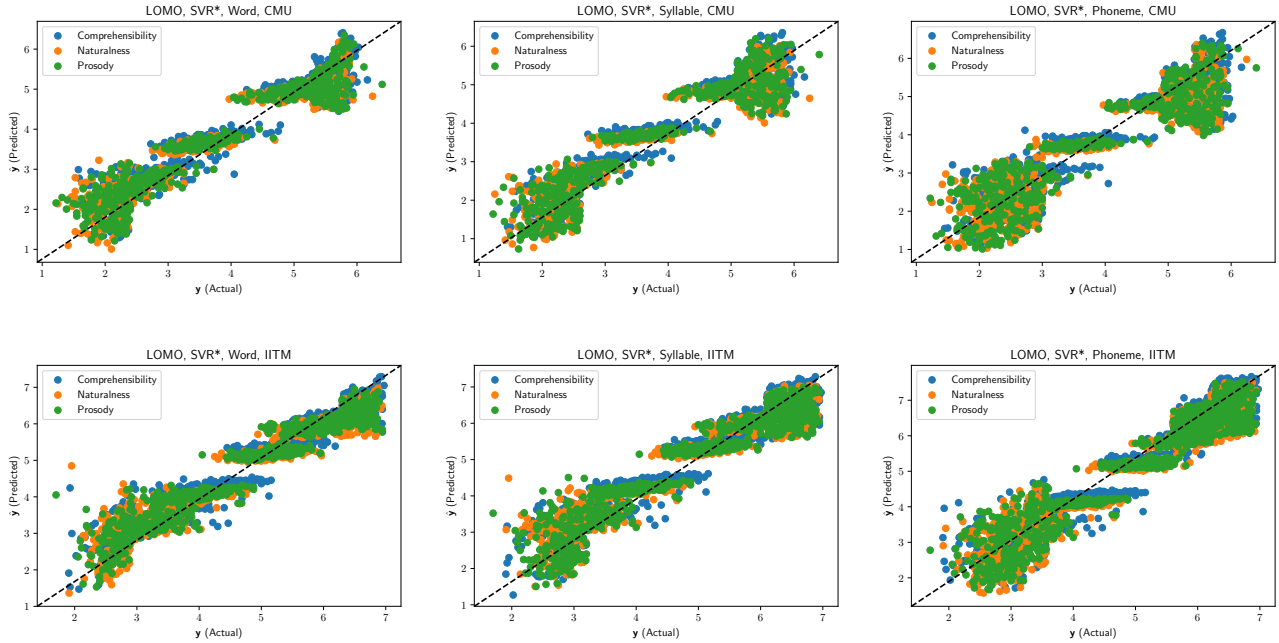


Figure 6.11 Scatterplots for LOMO-CV performed separately for CMU & IITM shown in two rows correspond to three feature-types Word, Syllabe and Phoneme respectively.

6.11 shows the results of SVR* graphically on various subjective-properties for both datasets. It can be seen from the plots, HMM and ORIG exhibit a comparable error to others. Furthermore, we observed that the speech signal with low-quality, e.g. HMM, are generally over-estimated, whereas high-quality, e.g. ORIG, speech tends to be under-estimated, as shown in Figure 6.11.

Comparison with other non-intrusive assessment metrics

We also compare the performance of LBOE with Quality-Net [180] and MOSNet [181]: recent deep learning-based non-intrusive assessment metrics. Quality-Net evaluates the quality of a synthesised speech based on the frame-level assessment. On the other hand, MOSNet uses a weighted score of utterance-level and frame-level. For fair comparison among LBOE, MOSNet and Quality-Net, we perform the quality prediction task to evaluate the same set of TTS systems under LOMO-CV setup. This experiment shows the robustness of the proposed LBOE evaluation method with these deep learning-based non-intrusive evaluation methods.

Table 6.16 shows the results of Quality-Net and MOSNet compared to the average of word-level LBOE evaluation on the selected TTS systems trained on both CMU and IITM datasets. We observe that the performance of Quality-Net and MOSNet is poor than the LBOE under LOMO-CV setup, which shows the usability and robustness of the proposed evaluation model. Additionally, it is also evident that the models perform better on IITM based TTS systems than the CMU.

Table 6.16 Comparison of LBOE with other *non-intrusive methods* under **LOMO-CV** criteria of model assessment.

Evaluation	TTS	CMU			IITM		
Metrics	Models	Quality-Net	MOSNet	LBOE	Quality-Net	MOSNet	LBOE
$\bar{R}_{LOMO}$	HMM	-0.13	0.00	**0.36**	0.02	0.04	**0.43**
	CLU	0.02	0.06	**0.72**	0.09	0.13	**0.72**
	DNN	0.16	0.18	**0.64**	0.11	0.21	**0.56**
	USS	0.08	0.13	**0.73**	0.16	0.17	**0.66**
	ORIG	-0.09	-0.02	**0.26**	0.03	0.01	**0.42**
$\bar{E}_{LOMO}$	HMM	1.57	1.34	**0.87**	1.21	0.93	**0.80**
	CLU	0.95	0.92	**0.32**	0.91	0.81	**0.31**
	DNN	0.96	0.89	**0.31**	0.87	0.76	**0.26**
	USS	0.99	0.93	**0.33**	0.92	0.86	**0.23**
	ORIG	1.29	1.13	**1.01**	1.13	0.98	**0.76**

6.6 Discussion

6.6.1 Characterisation of synthesised speech based on LLD feature-set

We compare the evaluation performance of our proposed feature-set on the basis of classification results with various state-of-the-art parameter sets proposed earlier during the series of Interspeech Challenges held in 2009 (IS09Emo) [324],2010 (IS10Paral) [325], 2011 (IS11SS) [326], 2012 (IS12ST) [327] and computational paralinguistics feature-set (IS13ComParE) [328], (ComParE16) [321] as well as (AVEC13) [329], (GeMAPS) and (eGeMAPS) [330].

On analysing Table E.1 and E.2, we observed the characteristics of LLD features in distinguishing the synthesised speech generated through various TTS-models. The value of loudness is higher in ORIG, USS and lower for HMM model. The fundamental frequency (pitch) is supposed to be balanced for natural voice as in the case of ORIG, and for USS its value is in medium-range, lower for HMM, while higher in CLU and DNN model [333]. In contrast, the ZCR value is found to be lower in quality speech, e.g. ORIG, USS. The psychoacoustic sharpness is observed medium for the case of ORIG, USS and higher for HMM and lower for CLU, DNN model. Spectra Band Energy should be higher in lower frequency and balanced in higher frequency-range for a natural voice. Spectral Roll of Points are found to be medium in the case of ORIG, USS voice, higher for CLU models and lower for DNN and HMM models. Other spectral features also show similar behaviour.

MFCC features also have sufficient distinguishing capabilities in identifying the best TTS models [334]. MFCC(1-10) are mostly medium or medium-lower in characterising a good speech. While MFCC(11-16) are found to be medium or medium-higher for the same, a similar trend can be observed on both CMU and IITM

based TTS-models. For the HMM & DNN based models, MFCC features are mostly medium or medium-higher range compare to others while CLU models record lower-range values for MFCC.

Analysing the voicing related features, all models are very close to each other in terms of jitter values. However, looking at features local to a speech signal, HMM models receive high jitter values, CLU and DNN get low, while USS and ORIG register in-between values. In contrast on Shimmer, logHNR and Probability of Voicing based features, USS and ORIG models get values in the medium-range, higher for HMM, while lower for CLU and DNN models [335].

Comparison of LBOE with the subjective and objective evaluation methods

Section 6.5 shows that the TTS systems' subjective evaluation requires a significant human effort and time. Additionally, the variability in the listeners' response in the form of ACR and WER output does not favour the use of subjective evaluation. On the contrary, the objective evaluation matrices show inconsistent performance in the objective assessment of synthesised speech which requires a "golden" reference as a constraint.

On the other hand, the proposed Learning-Based Objective Evaluation framework has not only provided a robust alternative to them but also shown comparable performance to the current state-of-the-art of non-intrusive quality assessment models. It would be useful for researchers working in the area of speech synthesis evaluation.

CV-setup

The motivation to use Cross-Validation is mainly to avoid the results of being overfitted and to assess the generalised performance of the assessment-models. The question might be asked which CV-setup identifies the *best* assessment-model, LOTO or LOMO. Both try to evaluate the prediction models in different ways. LOTO-CV is suitable for assessing a large number of signals collectively. The principal aim here is to generalise the *ratings* of samples between several tests generated from whatever TTS model.

On the other hand, predicting the ratings of a completely new TTS configuration seems to be a challenging goal, although it can be achieved as described in Section 6.4. It should be noted that, in general, the CV assumes that training and testing data come from the same population. Hence for the LOMO-CV, new TTS signals predictability depends on its similarity with all TTS signals belonging to the training samples. The default CV-setup is based on the K-fold partitioning, which investigates each random split in a Monte-Carlo fashion [336].

Linear Vs. nonlinear quality-assessments models

It is evident from the results that the nonlinear models are the better choice for the assessment. The modelling of signal parameters is seen to be effective here via RBF-Kernel. The overall performance gap between linear and

nonlinear models is approximately (∼15-30%). They have been incorporated here to show how the correlation improves on an explicit nonlinearity of LLD-based features shown theoretically and empirically in Section 6.4 & 6.5.2, respectively. The model assessment results clearly identify that nonlinear modelling is most suitable for good prediction accuracy when such rigid validation conditions are applied, e.g. CV, usage of multiple databases, and no manual deletion of outliers. This confirms all attempts in the NiQA literature which emphasise the integration of all significant nonlinearities, e.g. ITU-T Recommendation P.563 [337], LCQA [338] or ANIQUE [339]. In contrast to these works, the LLD-based learning method introduced in this work delivers a time-efficient *generic* NiQA model. In addition, its transparent structure is a key asset that links explicitly with the physical property.

Though the nonlinear models are found to be superior in the study, linear models remain to be the best starting point when working on sparse empirical data. They can suggest valuable information about the modelling, e.g. the required degree of nonlinearity.

Validity & scope of application

This chapter proposes a set of LLD feature-set suitable for evaluating synthesised speech sourced from various TTS-models. The instrumental assessment models considered in this study should be recognised as learning-based evaluation models. The generalisation capability of such models depends on how sufficiently rich training data is with respect to statistical *sample size*. It has been demonstrated during the discussion of LOTO-CV and LOMO-CV in Section 6.5.2.

We have trained five TTS models on both CMU and IITM datasets to acquire sufficient variations in the synthesised speech to be evaluated. All the TTS-models are independently assessed on word, syllable and phoneme-level LLD features. Beyond the CV-setup, the speech signals are supposed to be sufficiently long in order to level-out the variations induced in the spoken *sentence*. It is necessary as we have considered only acoustical properties for the comparison, which does not bother whether the text is spoken phonetically "correct" or not.

However, lexical properties are required to be included in the model training for explicit learning of phonetic correctness with the statistical perspective, but its effective use would need much more training data than the data used in the current study. Hence, we have not considered lexical properties. We believe that true analytical models are more complex and unrealistic for small-scale problems. Furthermore, in terms of entirely true evaluation measures, the instrumental model can probably never take over full-scale subjective auditory tests of the synthetic speech. But, such statistical learning-based models can be a time-efficient alternative that provides helpful diagnostic information.

6.7 Summary

This chapter has explored the area of speech synthesis, the last component in the SDS pipeline for converting the system's natural language response to speech waveform. In addition, it has also presented a novel framework for quality assessment *Learning-Based Objective Evaluation (LBOE)* which validates its capability by a comprehensive discussion on the evaluation of various text-to-speech synthesis systems.

We begin with the discussion on various TTS systems used for generating the speech materials for the experiments. For that, the chapter gives a brief note on several TTS engines, i.e. USS, HMM, CLU and DNN, used in the study of their quality assessment. To train the TTS engines, we utilise two publicly available speech datasets, IITM and CMU. The most natural synthesised speech will represent the system response in live conversation with the user in real-time.

First, a comparison of multiple speech synthesisers is made through various traditional objective and subjective evaluation measures in order to observe their usability and setbacks. Then, the proposed Learning-Based Objective Evaluation is presented, which acquires the positive points of both subjective and objective evaluation measures and also nullifies their negative merits. The Learning-Based Objective Evaluation method not only finds out the minimal set of LLD features that influence the TTS performance most but also scores each TTS model individually in a non-intrusive way.

Based on the minimal set of perceptually salient acoustic-features (LLDs), the framework builds quality-prediction models to evaluate various speech synthesis models. The following conclusions are drawn from the evaluation done through the CV-setup:

(i) Modelling the quality-assessment process of synthesised speech is possible as an alternative to costly and time-consuming subjective testing. The best (CV) performance, is observed as correlation of $\overline{R} = 0.87$ with $\overline{E} = 0.67$ in the CMU dataset, $\overline{R} = 0.90$ with $\overline{E} = 0.57$ in the IITM dataset in LOTO-CV assessment while $\overline{R} = 0.79$ with $\overline{E} = 0.32$ in CMU as well as $\overline{R} = 0.79$ with $\overline{E} = 0.33$ for IITM under LOMO-CV assessment configuration.

(ii) Regression-based quality-prediction models are robust and reliable with respect to the features at different levels, e.g. word, syllable and phoneme, being generated through different TTS models.

(iii) It is observed that the selected feature-set has the capability to capture the quality-effect from the synthesis speech. It provides an alternative for quality-prediction to conventional approaches.

(iv) A nonlinear model of regression performs better in predicting synthetic speech quality.

Hence, the final product would be an assessment framework that not only exhibits a set of selected acoustic features but also shows its performance through quality prediction and validation of synthesised speech.

Chapter 7

Limitations and Future Scope

This chapter concludes the thesis by summarising the contributions and marking out the limitations & future directions of the current work. The possibility of incorporating the current work to other Indic languages, i.e. Bengali (Bn), Telugu (Te), Gujarati (Gu), Tamil (Ta), Oriya (Or) and Punjabi (Pa) etc., has also been discussed in this chapter.

7.1 Thesis Summary & Contributions

The thesis has examined the challenges of developing a conversational system built upon native *Indian languages* for a real-world task. Following the modular architecture, the overall goal is to build a *data-driven* dialogue system with the ability to get improved over time and perceived as behaving human-like by the users. The components of a modular Spoken Dialogue System (SDS), i.e. ASR, SLU, DST, DM, NLDG and TTS, are based on statistical methods such as probabilistic distribution, neural network models, which allow them to handle both language-specific as well as language-independent uncertainties in both their input and their output.

The original contributions of this thesis include: the development of an *HDRS* corpus on which various state-of-the-art SLU and DST models, i.e. *NBT*, *GLAD*, *GCE*, *GSAT*, *Simple-BERT* and *SUMBT*, are implemented and compared; the RNNLG models, i.e. *H-LSTM*, *SC-LSTM*, *MSC-LSTM* and *ENC-DEC*, have been experimented and used to train corpus-based NLDG module on a self-collected corpus *dialogue-act & sentence* pairs without any alignment and annotations in an Indic language Hindi; construction of dialogue policy with RL based approaches, i.e. GP-SARSA, DQN, A2C, on the user-system act pairs generated by a user simulator; proposing a novel framework *LBOE* for quality assessment of a synthesised speech generated from various TTS engines, i.e. USS, HMM, CLU and DNN.

Following the modular architecture to build an SDS in Indic language, we work on the development of each component separately and resolve the challenges in combining them in a single SDS pipeline design later. Developing an SDS for a new language and new domain draws a great challenge to not only explore the existing

systems but also build a framework from scratch to perform the experiments. Based on this, the contributions of the thesis are summarised in the following sections.

7.1.1 Contribution to SLU & DST

In this thesis, a Hindi Dialogue Restaurant Search (HDRS) corpus is introduced to promote the research and development of a dialogue system for the Indic languages. It helps in designing the language understanding and state tracking modules for the Hindi language spoken dialogue system. The corpus comprises 1.4k fully-labelled hand-written dialogues collected using Wizard-of-Oz paradigm. In the corpus, there are 40% of the dialogues where the user changes her goal. Hence the corpus contains sufficient dialogue scenarios that are more natural and challenging for the dialogue state tracking.

As Hindi contains lots of lexical/morphological ambiguities, it becomes a key challenge for DST models to detect the DAs appropriately and keep the dialogue state updated. We have evaluated the performance of baseline SLU/DST and recent state-of-the-art neural belief trackers on the corpus. The Category-1 DST models, i.e. NBT-{CNN/DNN}, GLAD, GCE and GSAT, are trained using pre-trained embeddings such as GloVe, Word2Vec-{CBOW,SG}, FastText-{CBOW,SG}. On the other hand, the Category-2 DST-models, i.e. Simple-BERT DST, SUMBT, use the pre-trained multilingual-BERT encoder to handle dynamic ontology.

The data-driven approaches used in the Category-1 belief trackers depend entirely on the semantic quality of the underlying word vector space. Hence, the major part of this module investigated the problem of *semantic specialisation* by comparing the performance of various categories of word-level embeddings. Based on the experimental results, GSAT outperforms all other models with joint accuracy of 83.25%, followed by SUMBT. All category-1 DST models show better performance when trained on *FastText-SG*. Category-2 models can further be improved using the BERT explicitly trained on a Hindi corpus. By using semantically-induced embeddings, the performance of category-1 models can be improved.

Handling morphological properties is one of the major issues in building natural language processing applications in the Hindi language. The fact is proved by an experiment (see Section 3.6) where the GSAT-DST model is trained on both Hindi & English corpora with and without language-specific pre-trained embeddings. A significant difference is observed in the joint-goal accuracy in both cases when experimenting on the Hindi corpus compared to the English one. It shows the significance of FastText embedding in the NLP task of SLU/DST on the morphological languages, i.e. Hindi.

7.1.2 Decision Making in Dialogue through Reinforcement Learning

RL algorithms which follow a general POMDP framework are used in the thesis. In policy learning, they require a distribution over the dialogue states, also called the *belief state*, is to be maintained through the dialogue. Using

the learned dialogue policy, the dialogue manager selects an action based on the current belief state. The process of learning the optimal policy is called *policy optimisation*.

Reinforcement learning approaches are practical for decision making and provide a general framework for designing automatic dialogue policy learning without relying anymore on the hand-crafted rules. For this part, the thesis discusses the criteria for reward estimation and shows the comparative performance of the value-based e.g. GP-SARSA, DQN and policy-based, i.e. gradient-based RL methods in modelling the policy. We found that an advantage function when applied as a critic policy with the gradient-based (a type of policy-based function) RL approach, significantly improves the dialogue performance.

7.1.3 Contribution to Hindi Dialogue Generation

This part has shown that with the right architecture design, an RNN based language generation model can produce high-quality dialogue responses learning from human-authored examples in an Indic language. In contrast to previous approaches that rely on constructing intermediate representations with explicit linguistic annotations, the RNNLG models generate these representation without relying on any annotations and avoid unused and redundant ones to improve the learnability and scalability of the NLDG component. As per the discussion in Section 2.1, this backward integration of an NLDG component is proven to be effective in the modular architecture of a spoken dialogue system because it helps mitigate the development load and be competitive in terms of human-perceived quality over the previous approaches.

The RNNLG framework has been adapted to explore and construct various RNN models for generating responses for a Hindi dialogue system. The general architecture of the RNNLG framework is a combined process of *sentence-planning* and *surface-realisation* which enables training corpus-based NLDG on *dialogue-act & sentence pairs* without any alignment annotations. Alternatively, these alignments are learned as sentence planning either by gating or attention mechanism, while the surface-realisation is achieved by a recurrent structure like RNN or LSTM.

Based on the gating mechanism, the sentence-planning has been investigated by three architectures of different capabilities (a) heuristically-gated models (H-RNN, H-LSTM), (b) semantically-controlled models (SC-RNN, SC-LSTM, MSC-LSTM) and (c) ENC-DEC. Several existing baselines, e.g. HDC, n-gram and KNN, are used for comparison. HDC and KNN models generate rigid utterances, while class-based n-gram and RNNLG based models have the ability to generate novel utterances based on the probability distribution of tokens in the training data. In terms of BLEU-score, T-Error and S-Error, RNNLG based models have shown better performances. MSC-LSTM is the best architecture among all the models due to its ability to remember key-phrases corresponding to DA-type and slot-value pairs by incorporating DA differently from the SC-LSTM architecture.

In contrast, H-LSTM does not perform well for the binary slot-value pairs, and the ENC-DEC model allows the repetition of slot-value information.

To further enhance the performance of RNNLG models, an adaptation recipe of delexicalisation is utilised to facilitate training under a limited data scenario. In this method, all sentences are pre-processed via delexicalisation, where slot-valued specific words are replaced with their corresponding generic tokens based on the ontology. An RNNLG model's output, a sequence of tokens, further needed to be lexicalised for the appropriate surface realisation.

7.1.4 Contribution to TTS & Quality Assessment

As a last component in the SDS pipeline, a TTS system plays a significant role in making the conversation with a spoken dialogue system more natural and human-like. To achieve this, we have not only trained various "off-the-shelf" TTS systems: Unit selection speech synthesis (USS) [34], Hidden Markov Model speech synthesis (HMM) [145], Clustergen speech synthesis (CLU) [147] and Deep Neural Network-based speech synthesis (DNN), i.e. Tacotron-2 [38], but also compared the quality of speech they produce through.

The thesis proposes *Learning-Based Objective Evaluation (LBOE)*, a novel framework for quality assessment, and validates its capability by a comprehensive discussion on the evaluation of various text-to-speech synthesis systems. First, a comparison of multiple speech synthesisers is made through various traditional objective and subjective evaluation measures to observe their usability and setbacks. Learning-Based Objective Evaluation method not only finds out the minimal set of LLD features influence the TTS performance most but also scores each TTS model individually in a non-intrusive way.

The proposed Learning-Based Objective Evaluation acquires the positive points of both subjective and objective evaluation measures and also nullify their negative merits. It provides evaluation results as reliable and accurate as through the subjective evaluation with only one-time manual and financial support. Like the objective evaluation, it relies only on the speech files of several categories without entirely depending on the availability of original (human) speech files. In comparison to the recent state-of-the-art of deep-learning-based NiQA models, i.e. Quality-Net [180], MOSNet [181], our framework is shown to be more robust and accurate.

Based on the minimal set of perceptually salient acoustic-features (LLDs), the framework builds quality-prediction models in order to evaluate various speech synthesis models. It provides a low cost and less time-consuming alternative to modelling the quality-assessment.

7.1.5 Other Indic Languages

We show that our system handles the complex morphology posed by the Hindi language. However, its downstream performance shows a greater dependence on high-quality word vectors, especially for low-frequency

words. Other Indic languages are also morphologically rich, follow the same sentence structure. To exemplify it, an utterance in Hindi is translated to some Indic languages, i.e. Bengali (Bn), Telugu (Te), Gujarati (Gu), Tamil (Ta), Oriya (Or) and Punjabi (Pa) etc., with explicit sentence-structure annotation as below:

(En). [I]$_S$ [am searching]$_V$ [for Bengali]$_{O_m}$ [food]$_O$ [in the southern part of the city]$_{C_m}$.

(Hi). [मैं]$_S$ [शहर के दक्षिणी हिस्से में]$_{C_m}$ [बंगाली]$_{O_m}$ [खाना]$_O$ [ढूंढ रहा हूँ]$_V$।

(Bn). [আমি]$_S$ [শহরের দক্ষিণ অংশে]$_{C_m}$ [বাংলা]$_{O_m}$ [খাবারের]$_O$ [জন্য খোঁজ করছি]$_V$।

(Te). [నేను]$_S$ [నగరం యొక్క దక్షిణ భాగంలో]$_{C_m}$ [బెంగాలీ]$_{O_m}$ [ఆహారం]$_O$ [కోసం చూస్తున్నాను]$_V$.

(Gu). [હું]$_S$ [શહેરના દક્ષિણ ભાગમાં]$_{C_m}$ [બંગાળી]$_{O_m}$ [ખોરાક]$_O$ [શોધી રહ્યો છું]$_V$.

(Ta). [நான்]$_S$ [நகரத்தின் தெற்கு பகுதியில்]$_{C_m}$ [பெங்காலி]$_{O_m}$ [உணவைத்]$_O$ [தேடுகிறேன்]$_V$.

(Or). [ମୁଁ]$_S$ [ସହରର ଦକ୍ଷିଣ ଭାଗରେ]$_{C_m}$ [ବଙ୍ଗାଳୀ]$_{O_m}$ [ଖାଦ୍ୟ]$_O$ [ଖୋଜୁଛି]$_V$ ।

(Pa). [ਮੈਂ]$_S$ [ਸ਼ਹਿਰ ਦੇ ਦੱਖਣੀ ਹਿੱਸੇ ਵਿੱਚ]$_{C_m}$ [ਬੰਗਾਲੀ]$_{O_m}$ [ਭੋਜਨ]$_O$ [ਲੱਭ ਰਿਹਾ ਹਾਂ]$_V$।

Assuming S_m as a subject modifier, O_m as object modifier, V_m as expected verb post-modifiers and C_m as the optional verb post-modifiers, used as case markers for depicting the sentence structure. In general, Indic languages follow a common sentence structure of SOV (Here, S=Subject, O=Object and V=Verb) in contrast to English which is based on SVO structure. In addition, case markers are postpositioned and are strongly bound to nouns. Similar challenges are visible in all Indic languages. Therefore, the statistical models with little improvement would suffice to build a full-fledged dialogue system in an Indic language.

7.2 Limitations & Future Directions

All the data-driven neural network-based models, i.e. NBT, GLAD, GCE, GSAT, SUMBT, Simple-BERT, explored in this thesis, do not rely much on in-domain semantic lexicons. They are, therefore, supposed to be scalable to larger and more sophisticated domains. However, as we scale up the training process with complex data, these network-based models sometimes lead to unexpected failures, which is not desirable in real-time applications. Analysis of the semantic details acquired from the pre-trained embeddings, FastText, GloVe, BERT etc., may provide a relatively elegant way for the system designer to circumvent and remove some frequently occurring errors. It still remains to be observed how robust our SLU/DST models perform when deployed in a real-world scenario and how much harder it gets to deal with such surprising failures.

The presented work has explored and revealed the dependency of language understating performance on modelling the Morphology, Code-Mixing, Echo-Words, Lexical Variations. The developed system was exposed to these complexities through some language-specific pre-trained vectors. The experiments have proved that for morphologically rich languages (Indic languages) such as Hindi, the performance of language understanding was improved when the underlying vector space was transformed to model language-specific morphology. However, if this design framework is applied to other non Indo-European languages, such as Arabic, Chinese, Japanese or

Vietnamese, it would pose more substantial challenges due to their structural differences. For example, Chinese and Japanese are analytic and isolating (segment sentences rather than words) than the Indo-European, which are predominantly synthetic and inflected, while the Vietnamese represent the written tokens in the form of syllables. Building language understanding modules for these languages would present an exciting challenge.

Although the Category-1 DST models are found to be performing well, they can further be improved by using semantically induced in-domain word-embeddings. In comparison, Category-2 models are more robust towards new slot values, which is suitable for systems with dynamic ontologies. As the BERT model plays a crucial role in these models' performance, a BERT trained explicitly in the Hindi language is expected to deliver better performance.

The reward estimated by various approaches is used for dialogue evaluation in the thesis, which utilises the success information [207, 340]. Although the success information is a prominent feature for dialogue quality in task-oriented slot-filling SDSs, this is only one aspect of user satisfaction. It is thus an important area to work further to define and estimate the dialogue quality. The dialogue policies we have explored in this thesis mainly operate at the semantic level of abstraction 'dialogue acts' [341], that are human-engineered, and requires expert knowledge. Due to this, the system suffers from scalability issues because the quality and variability of a potential output response are highly dependent on the selection of dialogue acts. Recently, people have attempted to learn the latent dialogue acts implicitly, which does not need to define a list of dialogue acts in advance [342], and it might be helpful in generating a more diverse and natural system response.

Several limitations have been observed in the experiments to build the NLDG component in an SDS. One limitation is related to the process of delexicalisation [97, 99], which replaces domain-specific, i.e. ontology, words or phrases with placeholders (specialised tokens) to make the model's training easier. It is a rudimentary method as it relies on exact string matching. Due to this, the models are unable to produce complex linguistic phenomena, i.e. referential expressions or value-based comparisons. In addition, it may cause several ambiguities for large domains, creating scalability issues for the system. For example, for a restaurant domain, the word 'बंगाली' represents the food-type, but 'बंगाली' may also be used to depicts an ethnic group; hence the delexicalisation can be confusing in such cases. One possible solution to this is to apply the pointer network [343], which can resolve the issue by soft-string matching where the model use this technique to learn domain-specific words to match softly and replace them into sentences. Another limitation is regarding the evaluation of generated utterances; we are able to check only on the basis of syntactic error, which is not sufficient as the generated utterances may show different meanings.

In the quality assessment, a further research line can be opened related to the investigation of perceptual attributes that might better depict the physical properties of a synthesised speech. A deeper investigation of the psycho-physical characterisation and perceptual regularisation of synthetic speech would help validate the

quality description, and thus the overall performance could be enhanced. However, the proposed evaluation framework has presented a generalised architecture to use statistical approaches for the quality assessment. As the quality of the speech synthesis models is getting improved with recent advancements in the area, there is a constant need to explore and investigate the required features and speech characteristics to be included in the evaluation of TTS systems that could lead to an update of the LLD feature set.

Arguably the greatest bottleneck in developing a statistical dialogue system is collecting appropriate training data required to construct its components. It is especially true for the task-oriented dialogue systems, to which the availability of in-domain data is crucial plays a significant role for its optimal performance. For example, the task of SLU/DST relies on annotated corpora, which is based on the dialogue act taxonomies. The main limitation is that it requires experts to create accurate labels; and, therefore, hinders the collection process from being completely crowdsourced [58]. This also applies to other system modules such as SLU, NLDG and dialogue manager. Nevertheless, the Wizard-of-Oz approach has been used to build our dialogue corpora with coarse-grained annotations. It is much easier to run such crowdsourcing platforms as the collection procedure does not require expert knowledge.

We observe that the pipeline architecture has performed well in diagnosing and improving the components individually, but the improvement of a single module may not appropriately boost the overall performance of the integral system. Due to this, recent works are focussed on end-to-end approaches for building task-oriented dialogue systems [201, 111, 109, 344]. They aim to learn multiple components together without factorising the model into intermediate states that leads to avoid the need of intermediate labelling, and hence circumvents the major bottleneck of hand-crafting and expert-knowledge. Thus, these methods help potentially speed up the development process of the entire dialogue system and are worth further investigation.

In the current work, we have explored a unimodal natural-language based dialogue scenario. As the human-to-human conversation is multimodal, involving various linguistic forms and non-verbal signals [345], a multimodal human-to-computer conversation should therefore be more intuitive. Many researchers have explored multimodal scenarios in the conversational systems, such as Visual Question Answering (VQA) [346] based multimodal dialogue systems [347–352]. Essentially, the visual inputs, either in the form of image or video, provide rich information about the environment (User) in addition to the dialogue and help achieve good performance. The combined analysis and synthesis of language and vision may become the primary research focus in conversational systems in near future.

Acronyms

A2C Advantage Actor-Critic. xxi, 23, 70, 71, 84–86, 95, 97–100, 105, 163

A2CER Advantage Actor-Critic with Experience Replay. xxi, 7, 23, 71, 85, 86, 95, 98–106

ACR absolute category rating. 142, 156, 159

Adam The name is derived from adaptive moment estimation.. xxiii, 61, 99

ANIQUE Auditory Non-Intrusive QUality Estimation. 160

ANN Artificial Neural Network. 4

ANOVA ANalysis Of VAriance. 147, 148

API Application Programming Interface. 31

ASR Automatic Speech Recognition. v, 4, 5, 16–18, 22, 30, 31, 72–74, 96, 105, 163

ATIS Air Travel Information System. 19, 36, 37

AVEC13 Audio-Visual Emotion recognition Challenge (AVEC 2013). 140, 151–153, 159

B-LSTM Backward Long Short-Term Memory. xxvii

BERT Bidirectional Encoder Representations from Transformers. vi, xx, 8, 20, 35, 47, 56–59, 61, 62, 64–67, 163, 164, 168, 169

Bi-LSTM Bidirectional-LSTM. 53, 55, 56, 142, 152, 153

BLEU Bilingual Evaluation Understudy. xxiv, 91, 120–122, 124, 125, 127, 166

BSD Bark Spectral Distortion. 139, 140, 150

BV-Error Binary-Value Error. xxiv, 110, 120, 122, 125

CART Classification And Regression Tree. 26, 136

CBOW Continuous Bag-of-Words. xxiii, 60, 61, 63, 67, 164

CCG Combinatory Categorial Grammars. 19

CFG Context-Free Grammars. 19

CLID Cluster Identification Test. 27

CLU Clustergen speech synthesis. vii, xxii, 10, 25, 26, 131, 135, 136, 141, 147–151, 155, 156, 158, 159, 162, 164, 166

CLUNIT A vector of 25 features comprised of 24 $MFCC$s and a $F0$ is estimated for every 5ms in a speech. 26, 136

CMU Carnegie Mellon University. vii, xxii, xxiv, xxv, 25, 36, 37, 130–132, 134, 138, 140, 147–159, 161, 162

CNN Convolutional Neural Network. xx, xxvii, 20, 34, 47, 50–52, 61–67, 164

ComParE16 InterSpeech16 Computational Paralinguistics ChallengE Parameter Set. 159

CRF Conditional Random Field. 19

CTC Connectionist Temporal Classification. 4, 17

CV Cross-Validation. xxii, xxiv, xxv, xxix, 142, 143, 145–147, 154–158, 160–162

DA Dialogue-Act. v, vi, xx, 4, 6, 18, 33, 34, 38, 39, 41, 43, 45, 49, 62, 88, 107–115, 117–121, 123–125, 127, 164, 166

DBN Dynamic Bayesian Network. 19, 73

Deep-MOS predictor Deep MOS Predictor for Synthetic Speech Using Cluster-Based Modeling. 28

DM Dialogue Management. v, vi, 4, 5, 7, 16, 30, 163

DNN Deep Neural Network-based speech synthesis. vii, 10, 25, 26, 131, 141, 147–151, 155, 156, 158, 159, 162, 164, 166

DNN Deep Neural Network. 47, 51, 53, 61–67, 137, 164

DQN Deep Q-Network. xxi, 23, 70, 83, 95–97, 99, 100, 105, 163, 165

DRL Deep Reinforcement Learning. 70, 82, 84, 105, 106

DRT/MRT Diagnostic/Modified Rhyme Test. 27

DST Dialogue State Tracking. v, vi, xx, xxiii, 4–8, 10, 16, 19–22, 30, 31, 33–37, 39, 40, 47, 48, 53–57, 60–67, 72–75, 91, 163–165, 168–170

DSTC Dialogue State Tracking Challenges. 20, 36, 37

eGeMAPS extended Geneva Minimalistic Acoustic Parameter Set. 151–153, 159

ENC-DEC Encoder-Decoder. xxi, xxii, 24, 108, 109, 113, 118, 121, 122, 124–127, 163, 166

ER Experience-Replay. 70, 71, 85, 97

F-LSTM Forward Long Short-Term Memory. xxvii

F0 Fundamental frequency. 25, 26, 134, 136, 138, 139, 142, 149, 150

FastText Public library for efficient learning of word representations with subword information. vi, xxiii, 8, 60–63, 66, 67, 164, 165, 168

FT Public library for efficient learning of word representations with subword information. xxiii, 63

GCE Globally-Conditioned Encoder. xx, xxvii, 34, 47, 54–56, 61–67, 163, 164, 168

GeMAPS Geneva Minimalistic Acoustic Parameter Set. 151–154, 159

GLAD Global-Locally self-Attentive Dialogue State Tracker. xx, xxvii, 34, 47, 53–56, 61–67, 163, 164, 168

GloVe Global Vectors for Word Representation. vi, 8, 34, 60, 61, 63, 67, 164, 168

GMM Gaussian Mixture Model. 17

GP Gaussian Process. xxviii, 95, 96

GP-SARSA Gaussian-Process State-Action-Reward-State-Action. xxi, 23, 78, 82, 83, 95, 96, 99–101, 103–106, 163, 165

GRU Gated Recurrent Unit. 59, 62, 65

GSAT Global encoder and Slot-ATtentive decoder. xx, xxvii, 34, 47, 55, 56, 61–67, 163–165, 168

H-LSTM Heuristically-gated LSTM. xxi, xxii, 24, 108, 109, 113–115, 117, 119, 121, 123–127, 163, 166

H-RNN Heuristically-gated RNN. 108, 109, 114, 121, 123–125, 127, 166

H2H Human-to-Human. 36, 37

H2M Human-to-Machine. 36, 37

HDC Hand-Crafted. 25, 109, 120–122, 124, 125, 127, 166

HDRS Hindi Dialogue Restaurant Search. vi, xix, xxiii, 8, 10, 15, 21, 31, 37–40, 44, 66, 67, 71, 94, 102, 163, 164

HLD high-level-descriptors. 140

HMM Hidden Markov Model speech synthesis. vii, 10, 25, 131, 134, 135, 141, 147–151, 155–159, 162, 164, 166

HMM Hidden Markov Model. 4, 5, 17, 25, 26, 28, 134, 136

HTTP Hyper Text Transfer Protocol. 30

HTTPS Hyper Text Transfer Protocol Secure. 30

IITM Indian Institute of Technology Madras. vii, xxii, xxiv, xxv, 25, 130–132, 134, 138, 140, 147–154, 156–159, 161, 162

IS Important Sampling. xxviii, 85, 87, 98, 101

IS09Emo InterSpeech 2009 Emotion Challenge Parameter Set. 152, 153, 159

IS10Paral InterSpeech 2010 Paralinguistics Challenge Parameter Set. 152, 153, 159

IS11SS InterSpeech 2011 Speaker State Challenge Parameter Set. 152, 153, 159

IS12ST InterSpeech 2012 Speaker Trait Challenge Parameter Set. 152, 153, 159

IS13ComParE InterSpeech13 Computational Paralinguistics ChallengE Parameter Set. 140, 152, 153, 159

ITU International Telecommunication Union. 27

ITU-T ITU Telecommunication Standardization Sector. 28, 160

JSON Java Script Object Notation. 30

KNN K-Nearest Neighbors. 25, 108–110, 120–122, 124, 125, 127, 166

KVRET Key-Value Retrieval. 36, 37

lagDiff lag Difference. 139, 149, 150

LBOE Learning-Based Objective Evaluation. vii, xxii, xxv, 10, 129, 130, 137, 140–142, 150, 156–158, 160–162, 164, 166, 167

LCQA Low Complexity Quality Assessment. 160

LDA Linear Discriminant Analysis. 142, 152, 153

LLD low-level-descriptors. vii, 137, 140–143, 150, 159–162, 167, 170

LM Language Model. 23, 108

logHNR Logarithmic Harmonics-to-Noise Ratio. 142, 159

LOMO Leave-One-Model-Out. xxii, xxv, 129, 146, 147, 154–158, 160–162

LOTO Leave-One-Test-Out. xxii, xxiv, xxv, 146, 154–157, 160–162

LPC Linear Predictive Coding. 139

LPCC Linear Predictive Coding Coefficients. 28, 139, 149, 150

LSTM Long Short-Term Memory. xxix, 4, 17, 24, 28, 53, 59, 62, 65, 108, 109, 112–114, 116, 117, 119, 123, 127, 142, 152, 166, 171, 174, 176–178

M2M Machine-to-Machine. 36, 37

MC Monte Carlo. 83

MCD Mel-Cepstral Distortion. 28, 139, 149, 150

MDP Markov Decision Process. 22, 23, 72, 73, 75, 76, 87

METEOR Metric for Evaluation of Translation with Explicit ORdering. 91

MFCC Mel-Frequency Cepstral Coefficients. 17, 26, 28, 136, 142, 159

MLSA Mel Log Spectrum Approximation. 26, 136

MOS Mean Opinion Score. 28, 140, 172, 176

MOS-X MOS-Expanded. 138, 149

MOSNet MOSNet: Deep Learning based Objective Assessment for Voice Conversion. 28, 129, 157, 158, 167

MSC-LSTM Modified Semantically Controlled LSTM. xxi, 108, 109, 113, 117, 121, 123–128, 163, 166

MultiWOZ Multi-Domain Wizard-of-Oz. 20, 36, 37

NBT Neural Belief Tracker. 47, 51, 53, 61–67, 163, 164, 168

NiQA Non-intrusive Quality Assessment. 129, 160, 167

NLDG Natural Language Dialogue Generation. v, vi, 5–7, 9, 10, 13, 16, 23, 24, 30, 74, 107–110, 120, 123, 125, 126, 163, 165, 166, 169, 170

NLG Natural Language Generation. 128

NLP Natural Language Processing. 54, 60, 67, 165

NN Neural Network. 70, 104, 105

ORIG Original human speech files. 140, 141, 143, 150–152, 155–159

P.563 Single-ended method for objective speech quality assessment in narrow-band telephony applications. 28, 160

PCA principal component regression. 144

PESQ Perceptual Evaluation of Speech Quality. 28, 139, 140, 149, 150

PhD Doctor of Philosophy. ix, 137

PLS Partial Least Squares Regression. 144, 154, 156

POMDP Partially Observable Markov Decision Process. xxi, 23, 72, 73, 75, 77, 78, 82, 88, 91, 105, 165

Quality-Net Quality-Net: An End-to-End Non-intrusive Speech Quality Assessment Model based on BLSTM. 28, 129, 157, 158, 167

RBF radial-basis-function. 145, 152, 156, 160

RL Reinforcement Learning. vi, xxi, xxiii, xxviii, 5, 9, 22, 23, 70, 71, 74–76, 78, 80–83, 85, 87, 91, 93–101, 104, 105, 163, 165

RMSE Root-Mean-Square-Error. xxix, 139, 145, 146, 149, 150

RNN Recurrent Neural Network. xx, 10, 17, 24, 50, 51, 59, 62, 67, 108, 110–112, 114, 116, 123, 127, 165, 166, 174, 177

RNNLG Recurrent Neural Network Language Generation. vii, xxi, xxii, xxiv, 7, 9, 24, 25, 108–111, 113, 120–122, 126, 127, 163, 165, 166

RNNLM RNN based Language Modelling. 110

S-Error Slot Error. xxiv, 110, 120–122, 124, 125, 127, 166

s2s sequence-to-sequence. 26, 111, 114, 119, 127, 137

SARSA State-Action-Reward-State-Action. 83, 96

SC-LSTM Semantically Controlled LSTM. xxi, 24, 108, 109, 113, 115–117, 121, 123–127, 163, 166

SC-RNN Semantically Controlled RNN. xxi, 108, 109, 116, 121, 123–125, 127, 166

SDS Spoken Dialogue System. v, 1, 3–6, 8, 10, 11, 16–18, 24, 25, 29–31, 47, 67, 73–75, 78, 82, 91, 95, 96, 105, 107, 108, 126, 161, 163, 164, 166, 169

Self-Attn Self-Attention. 55, 56

SER Semantic Error Rate. xxiv, 90, 98, 100, 103, 105

SG Skip-gram. xxiii, 60, 61, 63, 67, 164

SGD Stochastic Gradient Descent. 49, 62

SGD Schema-Guided Dialogue. 36, 37

SILP Our Lab's name: Speech Image and Language Processing. ix, 7, 30, 177

SILPA SILPAssistant. xix, 7, 12, 14, 29–31

SL Supervised Learning. 94, 95, 103

SLU Spoken Language Understanding. v, vi, xxiii, 4, 5, 7, 8, 16, 18–22, 30, 31, 33–35, 37, 47–49, 60, 62, 66, 67, 72, 74, 91, 96, 163–165, 168, 170

SNR signal-to-noise-ratio. 139, 149, 150

SOV Subject Object Verb. 24, 168

SST Standard Segmental Test. 27

STC Semantic Tuple Classifier. 49

SUMBT Slot-Utterance Matching for universal and scalable Belief Tracking. xx, 35, 47, 58, 61, 62, 64, 65, 67, 163, 164, 168

SUS Semantically Unpredictable Sentences. 27

SV slot-value. xx, xxii, 48, 51, 53–55, 124, 125

SVM Support Vector Machine. xxii, xxix, 19, 49, 62, 142, 144, 152, 153

SVO Subject Verb Object. 24, 168

SVR Support Vector Regression. xxv, 144, 145, 154–157

T-Error Total Error. xxiv, 110, 120–122, 124, 125, 127, 166

TD Temporal-Difference. 78, 85, 86, 96

TTS Text-To-Speech. v, vii, xxii, xxiv, 5, 7, 9, 10, 16, 25, 26, 28, 30, 31, 74, 105, 129–131, 133–140, 142, 143, 145–164, 166, 167, 170

UAR unweighted average recall. 143

USS Unit selection speech synthesis. vii, 10, 25, 26, 131, 132, 134, 135, 141, 147–152, 155, 156, 158, 159, 162, 164, 166

V-LSTM Vanilla-LSTM. 109, 112, 113, 121, 123–125

VC Voice Conversion. 28

VoiceXML Voice Extensible Markup Language. 69

VQA Visual Question Answering. 170

W2V Public library for computing continuous distributed representations of words.. xxiii, 63

WER word error rates. 138, 148, 159

Word2Vec Public library for computing continuous distributed representations of words.. vi, xxiii, 8, 60, 63, 67, 111, 119, 164

WOZ Wizard-of-Oz. xix, xx, xxiii, 36–38, 41–43, 66, 67, 94, 105, 164, 170

WSS Weighted Spectral Slope. 139, 149, 150

XAVIER A random word-vector. 60, 63, 64

ZCR Zero Crossing Rate. 142, 159

References

[1] Shrikant Malviya, Rohit Mishra, Santosh Kumar Barnwal, and Uma Shankar Tiwary. HDRS: Hindi dialogue restaurant search corpus for dialogue state tracking in task-oriented environment. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 29:2517–2528, 2021. xxiii, 7, 63, 64, 66, 83, 91, 93

[2] Steve Young, Milica Gašić, Blaise Thomson, and Jason D Williams. POMDP-based statistical spoken dialog systems: A review. *Proceedings of the IEEE*, 101(5):1160–1179, 2013. 1, 4, 5, 26, 62, 63

[3] Alan M. Turing. Computing machinery and intelligence. *Mind*, 59(October):433–60, 1950. 1

[4] Joseph Weizenbaum. Eliza—a computer program for the study of natural language communication between man and machine. *Communications of the ACM*, 9(1):36–45, 1966. 1

[5] Bhuwan Dhingra, Lihong Li, Xiujun Li, Jianfeng Gao, Yun-Nung Chen, Faisal Ahmed, and Li Deng. Towards end-to-end reinforcement learning of dialogue agents for information access. In *Proceedings of ACL*, pages 484–495, 2017. 2

[6] Neil Patel, Sheetal Agarwal, Nitendra Rajput, Amit Nanavati, Paresh Dave, and Tapan S Parikh. Experiences designing a voice interface for rural india. In *2008 IEEE Spoken Language Technology Workshop*, pages 21–24. IEEE, 2008. 2

[7] Jonathan Lyons. Artificial stupidity. In *Proceedings of ACM SIGGRAPH*, page 27. Association for Computing Machinery, 2007. 2

[8] Stephanie Seneff. Response planning and generation in the mercury flight reservation system. *Computer Speech & Language*, 16(3-4):283–312, 2002. 3

[9] Teruhisa Misu and Tatsuya Kawahara. Speech-based interactive information guidance system using question-answering technique. In *Proceedings of ICASSP*. IEEE, 2007. 3

[10] Jingjing Liu, Scott Cyphers, Panupong Pasupat, Ian McGraw, and James Glass. A conversational movie search system based on conditional random fields. In *INTERSPEECH*, 2012. 3

[11] Jason D Williams. Applying pomdps to dialog systems in the troubleshooting domain. In *Proceedings of the Workshop on Bridging the Gap: Academic and Industrial Research in Dialog Technologies*, pages 1–8, 2007. 3, 81

[12] Tsung-Hsien Wen, Milica Gašić, Nikola Mrkšić, Pei-Hao Su, David Vandyke, and Steve Young. Semantically conditioned LSTM-based natural language generation for spoken dialogue systems. In *Proceedings of EMNLP*, pages 1711–1721, 2015. 3, 4, 22, 26, 96, 101, 102

[13] Tim Paek and Roberto Pieraccini. Automating spoken dialogue management design using machine learning: An industry perspective. *Speech communication*, 50(8-9):716–729, 2008. 3

[14] Dan Jurafsky and James H Martin. Chatbots and dialogue systems. In *Speech and Language Processing (3rd draft ed.)*. Stanford University, 2019. 3, 17, 26

[15] Roberto Pieraccini and Juan Huerta. Where do we go from here? research and commercial spoken dialog systems. In *Proceedings of SIGDIAL*, pages 1–10, 2005. 3, 11, 61, 64

[16] Herve A Bourlard and Nelson Morgan. *Connectionist speech recognition: a hybrid approach*, volume 247. Springer Science & Business Media, 2012. 4

[17] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997. 4, 16, 48, 99

[18] Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In *Proceedings of ICML*, pages 369–376, 2006. 4, 16

[19] Gokhan Tur and Li Deng. Intent determination and spoken utterance classification. *Spoken Language Understanding: Systems for Extracting Semantic Information from Speech*, pages 93–118, 2011. 4, 17

[20] Puyang Xu and Ruhi Sarikaya. Convolutional neural network based triangular crf for joint intent detection and slot filling. In *IEEE Workshop on ASRU*, pages 78–83. IEEE, 2013. 4, 17

[21] Yun-Nung Chen, William Yang Wang, and Alexander I Rudnicky. Unsupervised induction and filling of semantic slots for spoken dialogue systems using frame-semantic parsing. In *IEEE Workshop on ASRU*, pages 120–125. IEEE, 2013. 4, 17

[22] Grégoire Mesnil, Yann Dauphin, Kaisheng Yao, Yoshua Bengio, Li Deng, Dilek Hakkani-Tur, Xiaodong He, Larry Heck, Gokhan Tur, Dong Yu, et al. Using recurrent neural networks for slot filling in spoken language understanding. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 23(3): 530–539, 2014. 4, 17, 18

[23] John R Searle, PG Searle, S Willis, John Rogers Searle, et al. *Speech acts: An essay in the philosophy of language*, volume 626. Cambridge university press, 1969. 4, 16

[24] Nikola Mrkšić and Ivan Vulić. Fully statistical neural belief tracking. In *Proceedings of ACL*, pages 108–113, 2018. 4, 18, 30, 47, 55

[25] Ehud Reiter and Robert Dale. *Building Natural Language Generation Systems*. Cambridge University Press, New York, NY, USA, 2000. 4

[26] Kees van Deemter, Mariët Theune, and Emiel Krahmer. Real versus template-based natural language generation: A false opposition? *Computational linguistics*, 31(1):15–24, 2005. 4

[27] Alice H Oh and Alexander I Rudnicky. Stochastic language generation for spoken dialogue systems. In *ANLP-NAACL 2000 Workshop: Conversational Systems*, 2000. 4, 21, 96, 97, 107

[28] Amanda Stent, Rashmi Prasad, and Marilyn Walker. Trainable sentence planning for complex information presentation in spoken dialog systems. In *Proceedings of ACL*, ACL '04, 2004. 21, 95

[29] François Mairesse and Steve Young. Stochastic language generation in dialogue using factored language models. *Computational Linguistics*, 40(4):763–799, 2014. 4, 22, 95, 96

[30] Martin Sundermeyer, Ralf Schlüter, and Hermann Ney. Lstm neural networks for language modeling. In *INTERSPEECH*, 2012. 4

[31] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *Proceedings of ICLR*, 2015. 4, 105

[32] Ondřej Dušek and Filip Jurčíček. Sequence-to-sequence generation for spoken dialogue via deep syntax trees and strings. In *Proceedings of ACL*, pages 45–51, 2016. 4, 22, 96

[33] Hongyuan Mei, Mohit Bansal, and Matthew R. Walter. What to talk about and how? selective generation using LSTMs with coarse-to-fine alignment. In *Proceedings of NAACL*, pages 720–730, 2016. 4, 22, 96, 105

[34] Andrew J Hunt and Alan W Black. Unit selection in a concatenative speech synthesis system using a large speech database. In *Proceedings of ICASSP*, volume 1, pages 373–376. IEEE, 1996. 5, 23, 117, 119, 150

[35] Heiga Zen, Keiichi Tokuda, and Alan W Black. Statistical parametric speech synthesis. *speech communication*, 51(11):1039–1064, 2009. 5

[36] Aäron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio. In *Proceedings of 9th ISCA Speech Synthesis Workshop*, pages 125–125, 2016. 5, 24, 122

[37] Yuxuan Wang, R.J. Skerry-Ryan, Daisy Stanton, Yonghui Wu, Ron J. Weiss, Navdeep Jaitly, Zongheng Yang, Ying Xiao, Zhifeng Chen, Samy Bengio, Quoc Le, Yannis Agiomyrgiannakis, Rob Clark, and Rif A. Saurous. Tacotron: Towards end-to-end speech synthesis. In *INTERSPEECH*, pages 4006–4010, 2017. 24, 122

[38] Jonathan Shen, Ruoming Pang, Ron J. Weiss, Mike Schuster, Navdeep Jaitly, Zongheng Yang, Zhifeng Chen, Yu Zhang, Yuxuan Wang, RJ-Skerrv Ryan, Rif A. Saurous, Yannis Agiomyrgiannakis, and Yonghui Wu. Natural TTS synthesis by conditioning wavenet on MEL spectrogram predictions. In *Proceedings of ICASSP*, pages 4779–4783. IEEE, 2018. 5, 24, 122, 150

[39] Steve Young. Talking to machines (statistically speaking). In *Seventh International Conference on Spoken Language Processing*, 2002. 5, 17, 21

[40] Steve Young. Still talking to machines (cognitively speaking). In *INTERSPEECH*, 2010. 5, 11

[41] Bo Zhang, Qingsheng Cai, Jianfeng Mao, and Baining Guo. Planning and acting under uncertainty: A new model for spoken dialogue systems. In *Proceedings of the Seventeenth Conference on Uncertainty in Artificial Intelligence*, UAI'01, page 572–579, 2001. 5

[42] George Yule. *The study of language*. Cambridge University Press, 2020. 5, 19, 35

[43] Vishal Goyal and Gurpreet Singh Lehal. Hindi morphological analyzer and generator. In *2008 First International Conference on Emerging Trends in Engineering and Technology*, pages 1156–1159. IEEE, 2008. 5, 19, 35

[44] Shweta Vikram. Morphology: Indian languages and european languages. *International Journal of Scientific and Research Publications*, 3(6):1–5, 2013. 5, 19, 35, 59

[45] Dirk Geeraerts, Stefan Grondelaers, and Peter Bakema. *The structure of lexical variation: Meaning, naming, and context*, volume 5. Walter de Gruyter, 2012. 6, 19, 36

[46] Ananthakrishnan Ramanathan, Hansraj Choudhary, Avishek Ghosh, and Pushpak Bhattacharyya. Case markers and morphology: Addressing the crux of the fluency problem in english-hindi smt. In *Proceedings of ACL*, pages 800–808, 2009. 6, 19, 35

[47] Scott Miller, Robert Bobrow, Robert Ingria, and Richard Schwartz. Hidden understanding models of natural language. In *Proceedings of ACL*, pages 25–32, 1994. 6, 19, 36

[48] Shailendra Mohan. Echo-word formation in hindi. *Indian Linguistics*, 67:119–126, 2006. 6, 19, 36

[49] Amitava Das and Björn Gambäck. Code-mixing in social media text. the last language identification frontier? *Trait. Autom. des Langues*, 54(3):41–64, 2013. 6, 19, 35

[50] Dipam Goswami, Shrikant Malviya, Rohit Mishra, and Uma Shanker Tiwary. Analysis of word-level embeddings for indic languages on AI4Bharat-IndicNLP corpora. In *IEEE 8th UPCON*, pages 1–4. IEEE, 2021. 7

[51] Shrikant Malviya, Piyush Kumar, Suyel Namasudra, and Uma Shankar Tiwary. Experience replay based deep reinforcement learning for dialogue management optimisation. *ACM transactions of low-resource language information processing, Under-Review Since Sept 2021*, 2021. 8

[52] Sumit Singh, Shrikant Malviya, Rohit Mishra, Santosh Kumar Barnwal, and Uma Shanker Tiwary. Rnn based language generation models for a hindi dialogue system. In *International Conference on Intelligent Human Computer Interaction*, pages 124–137. Springer, 2019. 8, 33

[53] Shrikant Malviya, Santosh Kumar Barnwal, Rohit Mishra, and Uma Shankar Tiwary. A framework for quality assessment of synthesized speech using learning-based objective evaluation. *International Journal of Speech Technology*. 8

[54] Charles T Hemphill, John J Godfrey, and George R Doddington. The atis spoken language systems pilot corpus. In *Speech and Natural Language: Proceedings of a Workshop Held at Hidden Valley, Pennsylvania, June 24-27, 1990*, 1990. 11, 17, 19, 31, 32

[55] Jason D Williams. 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, 2012. 11

[56] Matthew Henderson, Blaise Thomson, and Jason D Williams. The third dialog state tracking challenge. In *2014 IEEE Spoken Language Technology Workshop (SLT)*, pages 324–329. IEEE, 2014. 11, 18, 32

[57] Anjuli Kannan and Oriol Vinyals. Adversarial evaluation of dialogue models. *arXiv preprint arXiv:1701.08198*, 2017. 11

[58] Matthew S Henderson. *Discriminative Methods for Statistical Spoken Dialogue Systems*. PhD thesis, University of Cambridge, 2015. 12, 14, 18, 43, 44, 153

[59] James Allen and Mark Core. Draft of damsl: Dialog act markup in several layers. Dagstuhl Workshop, 1997. [online] Available: https://www.cs.rochester.edu/research/cisd/resources/damsl/. 13

[60] James F Kurose. *Computer networking: A top-down approach featuring the internet, 3/E*. Pearson Education India, 2005. 15

[61] Lidia Mangu, Eric Brill, and Andreas Stolcke. Finding consensus in speech recognition: word error minimization and other applications of confusion networks. *Computer Speech & Language*, 14(4):373–400, 2000. 16, 80

[62] Mark Gales and Steve Young. The application of hidden markov models in speech recognition. *Found. Trends Signal Process.*, 1(3):195–304, 2007. 16

[63] Geoffrey Hinton, Li Deng, Dong Yu, George E Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N Sainath, et al. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal processing magazine*, 29(6):82–97, 2012. 16

[64] George E Dahl, Dong Yu, Li Deng, and Alex Acero. Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition. *IEEE Transactions on audio, speech, and language processing*, 20(1):30–42, 2011.

[65] Li Deng, Geoffrey Hinton, and Brian Kingsbury. New types of deep neural network learning for speech recognition and related applications: An overview. In *Proceedings of ICASSP*, pages 8599–8603. IEEE, 2013. 16

[66] Peter F Brown, Vincent J Della Pietra, Peter V Desouza, Jennifer C Lai, and Robert L Mercer. Class-based n-gram models of natural language. *Computational linguistics*, 18(4):467–480, 1992. 16

[67] Joshua T Goodman. A bit of progress in language modeling. *Computer Speech & Language*, 15(4): 403–434, 2001. 16

[68] Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan Černocký, and Sanjeev Khudanpur. Recurrent neural network based language model. In *INTERSPEECH*, 2010. 16, 98

[69] Xie Chen, Xunying Liu, Yongqiang Wang, Mark JF Gales, and Philip C Woodland. Efficient training and evaluation of recurrent neural network language models for automatic speech recognition. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 24(11):2146–2157, 2016. 16

[70] Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. Speech recognition with deep recurrent neural networks. In *Proceedings of ICASSP*, pages 6645–6649. IEEE, 2013. 16

[71] Awni Hannun, Carl Case, Jared Casper, Bryan Catanzaro, Greg Diamos, Erich Elsen, Ryan Prenger, Sanjeev Satheesh, Shubho Sengupta, Adam Coates, et al. Deep speech: Scaling up end-to-end speech recognition. *arXiv preprint arXiv:1412.5567*, 2014.

[72] Dzmitry Bahdanau, Jan Chorowski, Dmitriy Serdyuk, Philemon Brakel, and Yoshua Bengio. End-to-end attention-based large vocabulary speech recognition. In *Proceedings of ICASSP*, pages 4945–4949. IEEE, 2016. 16

[73] David R Traum. Foundations of rational agency, chapter speech acts for dialogue agents, 1999. 16

[74] Renato De Mori. Spoken language understanding: a survey. In *2007 IEEE Workshop on Automatic Speech Recognition & Understanding (ASRU)*, pages 365–376. IEEE, 2007. 17

[75] Sébastien Harispe, David Sánchez, Sylvie Ranwez, Stefan Janaqi, and Jacky Montmain. A framework for unifying ontology-based semantic similarity measures: A study in the biomedical domain. *Journal of biomedical informatics*, 48:38–53, 2014. 17

[76] Wayne Ward. The cmu air travel information service: Understanding spontaneous speech. In *Proceedings of the Workshop on Speech and Natural Language*, HLT'90, pages 127–129, 1990. 17

[77] Victor Zue, Stephanie Seneff, James R Glass, Joseph Polifroni, Christine Pao, Timothy J Hazen, and Lee Hetherington. Juplter: a telephone-based conversational interface for weather information. *IEEE Transactions on speech and audio processing*, 8(1):85–96, 2000. 17

[78] Mark Steedman. *The syntactic process*, volume 24. MIT press Cambridge, MA, 2000. 17

[79] Luke S Zettlemoyer and Michael Collins. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. *arXiv preprint arXiv:1207.1420*, 2012. 17

[80] Eugene Charniak. Statistical parsing with a context-free grammar and word statistics. *AAAI/IAAI*, 2005 (598-603):18, 1997. 17

[81] John M Zelle and Raymond J Mooney. Learning to parse database queries using inductive logic programming. In *Proceedings of the national conference on artificial intelligence*, Proceedings of AAAI, pages 1050–1055, 1996. 17

[82] Danqi Chen and Christopher D Manning. A fast and accurate dependency parser using neural networks. In *Proceedings of EMNLP*, pages 740–750, 2014. 17

[83] Christian Raymond and Giuseppe Riccardi. Generative and discriminative algorithms for spoken language understanding. In *INTERSPEECH*, 2007. 17, 64

[84] Yulan He and Steve Young. Spoken language understanding using the hidden vector state model. *Speech Communication*, 48(3-4):262–275, 2006. 17

[85] Rohit Kate and Raymond Mooney. Using string-kernels for learning semantic parsers. In *Proceedings of ACL*, pages 913–920, 2006. 18

[86] François Mairesse, Milica Gasic, Filip Jurcicek, Simon Keizer, Blaise Thomson, Kai Yu, and Steve Young. Spoken language understanding from unaligned data using discriminative classification models. In *Proceedings of ICASSP*, pages 4749–4752. IEEE, 2009. 18

[87] John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of ICML*, pages 282–289, 2001. 18

[88] Minwoo Jeong and Gary Geunbae Lee. Triangular-chain conditional random fields. *IEEE Transactions on Audio, Speech, and Language Processing*, 16(7):1287–1302, 2008. 18

[89] Kaisheng Yao, Baolin Peng, Yu Zhang, Dong Yu, Geoffrey Zweig, and Yangyang Shi. Spoken language understanding using long short-term memory neural networks. In *2014 IEEE Spoken Language Technology Workshop (SLT)*, pages 189–194. IEEE, 2014. 18, 29

[90] Edward Grefenstette, Phil Blunsom, et al. A convolutional neural network for modelling sentences. In *Proceedings of ACL*, 2014. 18

[91] Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. Teaching machines to read and comprehend. *Proceedings of NIPS*, 28: 1693–1701, 2015. 18

[92] William J Rapaport. Logical foundations for belief representation. *Cognitive Science*, 10(4):371–422, 1986. 18

[93] Jason Williams, Antoine Raux, Deepak Ramachandran, and Alan Black. The dialog state tracking challenge. In *Proceedings of SIGDIAL*, pages 404–413, 2013. 18, 19, 31, 32

[94] Matthew Henderson, Blaise Thomson, and Jason D. Williams. The second dialog state tracking challenge. In *Proceedings of SIGDIAL*, 2014. 18, 32, 40

[95] Seokhwan Kim, Luis Fernando D'Haro, Rafael E Banchs, Jason D Williams, and Matthew Henderson. The fourth dialog state tracking challenge. In *Dialogues with Social Robots*, pages 435–449. Springer, 2017. 18, 32

[96] Seokhwan Kim, Luis Fernando D'Haro, Rafael E Banchs, Jason D Williams, Matthew Henderson, and Koichiro Yoshino. The fifth dialog state tracking challenge. In *2016 IEEE Spoken Language Technology Workshop (SLT)*, pages 511–517. IEEE, 2016. 18, 32

[97] Matthew Henderson, Blaise Thomson, and Steve Young. Word-based dialog state tracking with recurrent neural networks. In *Proceedings of SIGDIAL*, pages 292–299, 2014. 18, 30, 43, 45, 152

[98] Lukas Zilka and Filip Jurcicek. Incremental lstm-based dialog state tracker. In *2015 Ieee Workshop on Automatic Speech Recognition and Understanding (Asru)*, pages 757–762. IEEE, 2015.

[99] Nikola Mrkšić, Diarmuid Ó Séaghdha, Blaise Thomson, Milica Gašić, Pei-Hao Su, David Vandyke, Tsung-Hsien Wen, and Steve Young. Multi-domain dialog state tracking using recurrent neural networks. In *Proceedings of ACL*, pages 794–799, 2015. 18, 30, 152

[100] Nikola Mrkšić, Diarmuid Ó Séaghdha, Tsung-Hsien Wen, Blaise Thomson, and Steve Young. Neural belief tracker: Data-driven dialogue state tracking. In *Proceedings of ACL*, pages 1777–1788, 2017. 18, 26, 30, 32, 33, 42, 43, 47

[101] Paweł Budzianowski, Tsung-Hsien Wen, Bo-Hsiang Tseng, Iñigo Casanueva, Stefan Ultes, Osman Ramadan, and Milica Gašić. MultiWOZ - a large-scale multi-domain Wizard-of-Oz dataset for task-oriented dialogue modelling. In *Proceedings of EMNLP*, pages 5016–5026, 2018. 18, 19, 31, 32

[102] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of NAACL*, pages 4171–4186, 2019. 19, 30, 51, 52

[103] Victor Zhong, Caiming Xiong, and Richard Socher. Global-locally self-attentive encoder for dialogue state tracking. In *Proceedings of ACL*, pages 1458–1467, 2018. 19, 30, 48, 55

[104] Elnaz Nouri and Ehsan Hosseini-Asl. Toward scalable neural dialogue state tracking. In *NeurIPS 2018, 2nd Conversational Al workshop*, 2018. 30, 55

[105] Vevake Balaraman and Bernardo Magnini. Scalable neural dialogue state tracking. In *2019 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pages 830–837. IEEE, 2019. 30, 50

[106] Guan-Lin Chao and Ian Lane. BERT-DST: Scalable end-to-end dialogue state tracking with bidirectional encoder representations from transformer. In *INTERSPEECH*, 2019. 30

[107] T. M. Lai, Q. Hung Tran, T. Bui, and D. Kihara. A simple but effective bert model for dialog state tracking on resource-limited systems. In *Proceedings of ICASSP*, pages 8034–8038, 2020. 30, 50

[108] Hwaran Lee, Jinsik Lee, and Tae-Yoon Kim. SUMBT: Slot-utterance matching for universal and scalable belief tracking. In *Proceedings of ACL*, pages 5478–5483, 2019. 19, 30, 52

[109] Tsung-Hsien Wen, David Vandyke, Nikola Mrkšić, Milica Gašić, Lina M. Rojas-Barahona, Pei-Hao Su, Stefan Ultes, and Steve Young. A network-based end-to-end trainable task-oriented dialogue system. In *Proceedings of EACL*, pages 438–449, Valencia, Spain, April 2017. 19, 26, 31, 32, 45, 153

[110] Layla El Asri, Hannes Schulz, Shikhar Sharma, Jeremie Zumer, Justin Harris, Emery Fine, Rahul Mehrotra, and Kaheer Suleman. Frames: a corpus for adding memory to goal-oriented dialogue systems. In *Proceedings of SIGDIAL*, pages 207–219, 2017. 32

[111] Mihail Eric, Lakshmi Krishnan, Francois Charette, and Christopher D. Manning. Key-value retrieval networks for task-oriented dialogue. In *Proceedings of SIGDIAL*, pages 37–49, 2017. 32, 153

[112] Pararth Shah, Dilek Hakkani-Tür, Gökhan Tür, Abhinav Rastogi, Ankur Bapna, Neha Nayak, and Larry P. Heck. Building a conversational agent overnight with dialogue self-play. *CoRR*, abs/1801.04871, 2018. 31, 32

[113] Abhinav Rastogi, Xiaoxue Zang, Srinivas Sunkara, Raghav Gupta, and Pranav Khaitan. Towards scalable multi-domain conversational agents: The schema-guided dialogue dataset. In *Proceedings of the AAAI*, volume 34, pages 8689–8696, 2020. 19, 31, 32

[114] Alan Ritter, Colin Cherry, and Bill Dolan. Unsupervised modeling of twitter conversations. In *Proceedings of NAACL*, pages 172–180, 2010. 19, 31, 32

[115] Ryan Lowe, Nissan Pow, Iulian Serban, and Joelle Pineau. The Ubuntu dialogue corpus: A large dataset for research in unstructured multi-turn dialogue systems. In *Proceedings of SIGDIAL*, pages 285–294, 2015. 32

[116] Saizheng Zhang, Emily Dinan, Jack Urbanek, Arthur Szlam, Douwe Kiela, and Jason Weston. Personalizing dialogue agents: I have a dog, do you have pets too? In *Proceedings of ACL*, pages 2204–2213, 2018. 19, 31, 32

[117] Bruce Lucas. VoiceXML for web-based distributed conversational applications. *Communications of the ACM*, 43(9):53–57, 2000. 20

[118] Stephen Sutton, David G Novick, Ronald Cole, Pieter Vermeulen, Jacques de Villiers, Johan Schalkwyk, and Mark Fanty. Building 10,000 spoken dialogue systems. In *Proceeding of ICSLP*, volume 2, pages 709–712. IEEE, 1996. 20

[119] David Goddeau, Helen Meng, Joseph Polifroni, Stephanie Seneff, and Senis Busayapongchai. A form-based dialogue manager for spoken language applications. In *Proceeding of ICSLP*, volume 2, pages 701–704. IEEE, 1996. 20, 61

[120] Staffan Larsson and David R Traum. Information state and dialogue management in the trindi dialogue move engine toolkit. *Natural language engineering*, 6(3-4):323–340, 2000. 20

[121] Oliver Lemon and Olivier Pietquin. Machine learning for spoken dialogue systems. In *INTERSPEECH*, 2007. 20, 61

[122] Simon Keizer, Milica Gasic, Filip Jurcicek, François Mairesse, Blaise Thomson, Kai Yu, and Steve Young. Parameter estimation for agenda-based user simulation. In *Proceedings of SIGDIAL*, pages 116–123, 2010. 62, 78, 79

[123] Lucie Daubigney, Matthieu Geist, Senthilkumar Chandramohan, and Olivier Pietquin. A comprehensive reinforcement learning framework for dialogue management optimization. *IEEE Journal of Selected Topics in Signal Processing*, 6(8):891–902, 2012. 69

[124] Milica Gašić and Steve Young. Gaussian processes for pomdp-based dialogue manager optimization. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 22(1):28–40, 2014. 20, 21, 61, 62, 69, 73, 84, 85, 92

[125] Esther Levin, Roberto Pieraccini, and Wieland Eckert. A stochastic model of human-machine interaction for learning dialog strategies. *IEEE Transactions on speech and audio processing*, 8(1):11–23, 2000. 20, 26, 62, 65, 78, 80

[126] Esther Levin, Roberto Pieraccini, and Wieland Eckert. Using markov decision process for learning dialogue strategies. In *Proceedings of ICASSP*, volume 1, pages 201–204. IEEE, 1998. 20, 21, 62

[127] Satinder Singh, Diane Litman, Michael Kearns, and Marilyn Walker. Optimizing dialogue management with reinforcement learning: Experiments with the njfun system. *Journal of Artificial Intelligence Research*, 16:105–133, 2002. 21, 64

[128] Jason D Williams and Steve Young. Partially observable markov decision processes for spoken dialog systems. *Computer Speech & Language*, 21(2):393–422, 2007. 21, 62, 63, 64, 81

[129] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013. 21, 84, 85

[130] Mehdi Fatemi, Layla El Asri, Hannes Schulz, Jing He, and Kaheer Suleman. Policy networks with two-stage training for dialogue systems. In *Proceedings of SIGDIAL*, pages 101–110, 2016. 21, 62, 84

[131] Marilyn A. Walker, Owen C. Rambow, and Monica Rogati. Training a sentence planner for spoken dialogue using boosting. *Computer Speech & Language*, 16(3):409 – 433, 2002. ISSN 0885-2308. Spoken Language Generation. 21, 95

[132] Irene Langkilde and Kevin Knight. Generation that exploits corpus-based statistical knowledge. In *Proceedings of ACL*, pages 704–710, 1998. 21, 95

[133] Verena Rieser and Oliver Lemon. Natural language generation as planning under uncertainty for spoken dialogue systems. In *Proceedings of EACL*, pages 683–691, 2009. 21, 95

[134] Irene Langkilde. An empirical verification of coverage and correctness for a general-purpose sentence generator. In *Proceedings of the international natural language generation conference*, pages 17–24, 2002. 21, 95, 96

[135] Amanda Stent, Matthew Marge, and Mohit Singhai. Evaluating evaluation methods for generation in the presence of variation. In *international conference on intelligent text processing and computational linguistics*, pages 341–351. Springer, 2005. 21

[136] François Mairesse, Milica Gašić, Filip Jurčíček, Simon Keizer, Blaise Thomson, Kai Yu, and Steve Young. Phrase-based statistical language generation using graphical models and active learning. In *Proceedings of ACL*, pages 1552–1561, 2010. 22, 95, 96

[137] Ioannis Konstas and Mirella Lapata. A global model for concept-to-text generation. *Journal of Artificial Intelligence Research*, 48:305–346, 2013. 22, 96

[138] Tsung-Hsien Wen and Steve Young. Recurrent neural network language generation for spoken dialogue systems. *Computer Speech & Language*, 63:101017, 2020. ISSN 0885-2308. 22, 96, 98, 100

[139] Rayner D'Souza, GV Shivakumar, D Swathi, and Pushpak Bhattacharyya. Natural language generation from semantic net like structures with application to hindi. In *Symposium on Translation Support Systems*, 2001. 22

[140] Shachi Dave, Jignashu Parikh, and Pushpak Bhattacharyya. Interlingua-based english–hindi machine translation and language divergence. *Machine Translation*, 16(4):251–304, 2001. 22

[141] Durgesh Rao, Kavita Mohanraj, Jayprakash Hegde, Vivek Mehta, and Parag Mahadane. A practical framework for syntactic transfer of compound–complex sentences for english–hindi machine translation. In *Proceedings of International Conference on KBCS*, pages 343–354, 2000. 22

[142] Mark Beutnagel, Alistair Conkie, Juergen Schroeter, Yannis Stylianou, and Ann Syrdal. The at&t next-gen tts system. In *Joint meeting of ASA, EAA, and DAGA*, pages 18–24. Citeseer, 1999. 23, 117

[143] Sathish Pammi, Marcela Charfuelan, and Marc Schröder. Multilingual voice creation toolkit for the mary tts platform. In *LREC*. Citeseer, 2010. 23, 118

[144] Marc Schröder and Anna Hunecke. Creating german unit selection voices for the mary tts platform from the bits corpora. *Proc. SSW6, Bonn, Germany*, 2007. 23, 118

[145] Heiga Zen, Keiichi Tokuda, and Alan W. Black. Statistical parametric speech synthesis. *Speech Communication*, 51(11):1039 – 1064, 2009. ISSN 0167-6393. 23, 119, 150

[146] Steve J. Young, D. Kershaw, J. Odell, D. Ollason, V. Valtchev, and P. Woodland. *The HTK Book Version 3.4*. Cambridge University Press, 2006. 23, 120

[147] Alan W Black. Clustergen: A statistical parametric synthesizer using trajectory modeling. In *INTERSPEECH*, pages 1762 – 1765, 2006. 23, 120, 150

[148] Alan W Black and Paul Taylor. Automatically clustering similar units for unit selection in speech synthesis. In *Fifth EUROSPEECH*, pages 601–604, 1997. 24, 121

[149] Keiichi Tokuda, Takao Kobayashi, Takashi Masuko, and Satoshi Imai. Mel-generalized cepstral analysis-a unified approach to speech spectral estimation. In *Third International Conference on Spoken Language Processing*, 1994. 24, 121

[150] Sercan Ö. Arık, Mike Chrzanowski, Adam Coates, Gregory Diamos, Andrew Gibiansky, Yongguo Kang, Xian Li, John Miller, Andrew Ng, Jonathan Raiman, Shubho Sengupta, and Mohammad Shoeybi. Deep voice: Real-time neural text-to-speech. In *Proceedings of ICML*, volume 70, pages 195–204, 2017. 24, 122

[151] Andrew Gibiansky, Sercan Arik, Gregory Diamos, John Miller, Kainan Peng, Wei Ping, Jonathan Raiman, and Yanqi Zhou. Deep voice 2: Multi-speaker neural text-to-speech. In *Proceedings of NIPS*, volume 30, pages 2962–2970, 2017.

[152] Wei Ping, Kainan Peng, Andrew Gibiansky, Sercan Ömer Arik, Ajay Kannan, Sharan Narang, Jonathan Raiman, and John Miller. Deep voice 3: Scaling text-to-speech with convolutional sequence learning. In *Proceedings of ICLR*, 2018. 24, 122

[153] Schuyler R Quackenbush, Thomas Pinkney Barnwell, and Mark A Clements. *Objective measures of speech quality*. Prentice Hall, 1988. 24

[154] Ulrnaturalnessich Heute. Speech-transmission quality: aspects and assessment for wideband vs. narrow-band signals. *Advances in digital speech transmission*, page 572, 2008. 24

[155] VJJP van Heuven, RL van Bezooijen, et al. Quality evaluation of synthesized speech. 1995. 24

[156] Sebastian Möller. *Quality Engineering: Qualität kommunikationstechnischer Systeme*. Springer-Verlag, 2017. 24

[157] RAMG van Bezooijen, VJ van Heuven, D Gibbon, R Moore, and R Winski. Assessment of synthesis systems. *Gibbon, D.; Moore, R.; Winski, R.(ed.), Handbook of standards and resources for spoken language systems*, pages 481–563, 1997. 24

[158] Ute Jekosch. *Voice and speech quality perception: assessment and evaluation*. Springer Science & Business Media, 2006. 24

[159] M Grice, K Vagges, and D Hirst. Prosodic form tests" and "prosodic function tests". *SAM final report*, 1992. 24

[160] Gerit P Sonntag and Thomas Portele. Purr—a method for prosody evaluation and investigation. *Computer Speech & Language*, 12(4):437–451, 1998. 24

[161] ITUT Rec. P. 85. a method for subjective performance assessment of the quality of speech voice output devices. *International Telecommunication Union, Geneva*, 1994. 24

[162] Mahesh Viswanathan and Madhubalan Viswanathan. Measuring speech quality for text-to-speech systems: development and assessment of a modified mean opinion score (mos) scale. *Computer Speech & Language*, 19(1):55–83, 2005. 24

[163] Catherine Mayo, Robert AJ Clark, and Simon King. Listeners' weighting of acoustic cues to synthetic speech naturalness: A multidimensional scaling analysis. *Speech Communication*, 53(3):311–326, 2011. 25, 26

[164] S. Möller, W. Chan, N. Côté, T. H. Falk, A. Raake, and M. Wältermann. Speech quality estimation: Models and trends. *IEEE Signal Processing Magazine*, 28(6):18–28, Nov 2011. ISSN 1558-0792. 25

[165] Yannis Stylianou and Ann K Syrdal. Perceptual and objective detection of discontinuities in concatenative speech synthesis. In *Proceedings of ICASSP*, volume 2, pages 837–840. IEEE, 2001. 25

[166] Jithendra Vepa and Simon King. Subjective evaluation of join cost and smoothing methods for unit selection speech synthesis. *IEEE Transactions on audio, speech, and language processing*, 14(5):1763–1771, 2006.

[167] Esther Klabbers, Jan PH Van Santen, and Alexander Kain. The contribution of various sources of spectral mismatch to audible discontinuities in a diphone database. *IEEE Transactions on Audio, Speech, and Language Processing*, 15(3):949–956, 2007.

[168] Paul Taylor. *Text-to-speech synthesis*. Cambridge university press, 2009. 25

[169] Arnd Mariniak. A global framework for the assessment of synthetic speech without subjects. In *Third EUROSPEECH*, 1993. 25

[170] Tiago H Falk and Sebastian Moller. Towards signal-based instrumental quality diagnosis for text-to-speech systems. *IEEE Signal Processing Letters*, 15:781–784, 2008. 25

[171] ITU-T Recommendation. Perceptual evaluation of speech quality (pesq): An objective method for end-to-end speech quality assessment of narrow-band telephone networks and speech codecs. *Rec. ITU-T P. 862*, 2001. 25

[172] Jing-Dong Chen and Nick Campbell. Objective distance measures for assessing concatenative speech synthesis. In *Sixth EUROSPEECH*, 1999. 25

[173] Milos Cernak and Milan Rusko. An evaluation of synthetic speech using the pesq measure. In *Proc. European Congress on Acoustics*, pages 2725–2728, 2005.

[174] Dong-Yan Huang. Prediction of perceived sound quality of synthetic speech. *Proc. APSIPA*, 2011.

[175] Florian Hinterleitner, Christoph Norrenbrock, Sebastian Möller, and Ulrich Heute. Predicting the quality of text-to-speech systems from a large-scale feature set. In *INTERSPEECH*, pages 383–387, 2013. 25

[176] Cassia Valentini-Botinhao, Junichi Yamagishi, and Simon King. Can objective measures predict the intelligibility of modified hmm-based synthetic speech in noise? In *INTERSPEECH*, 2011. 25

[177] Sebastian Möller, Florian Hinterleitner, Tiago H Falk, and Tim Polzehl. Comparison of approaches for instrumentally predicting the quality of text-to-speech systems. In *INTERSPEECH*, 2010. 25

[178] Christoph R Norrenbrock, Florian Hinterleitner, Ulrich Heute, and Sebastian Möller. Quality prediction of synthesized speech based on perceptual quality dimensions. *Speech Communication*, 66:17–35, 2015. 26

[179] Meng Tang and Jie Zhu. Text-to-speech quality evaluation based on lstm recurrent neural networks. In *2019 International Conference on Computing, Networking and Communications (ICNC)*, pages 260–264. IEEE, 2019. 26

[180] Szu wei Fu, Yu Tsao, Hsin-Te Hwang, and Hsin-Min Wang. Quality-net: An end-to-end non-intrusive speech quality assessment model based on blstm. In *INTERSPEECH*, pages 1873–1877, 2018. 26, 115, 141, 150

[181] Chen-Chou Lo, Szu-Wei Fu, Wen-Chin Huang, Xin Wang, Junichi Yamagishi, Yu Tsao, and Hsin-Min Wang. MOSNet: Deep Learning-Based Objective Assessment for Voice Conversion. In *INTERSPEECH*, pages 1541–1545, 2019. 26, 115, 141, 150

[182] Yeunju Choi, Youngmoon Jung, and Hoirin Kim. Deep MOS Predictor for Synthetic Speech Using Cluster-Based Modeling. In *INTERSPEECH*, pages 1743–1747, 2020. 26

[183] Tomoki Toda, Ling-Hui Chen, Daisuke Saito, Fernando Villavicencio, Mirjam Wester, Zhizheng Wu, and Junichi Yamagishi. The voice conversion challenge 2016. In *INTERSPEECH*, pages 1632–1636, 2016. 26

[184] Jaime Lorenzo-Trueba, Junichi Yamagishi, Tomoki Toda, Daisuke Saito, Fernando Villavicencio, Tomi Kinnunen, and Zhenhua Ling. The voice conversion challenge 2018: Promoting development of parallel and nonparallel methods. In *Proc. Odyssey 2018 The Speaker and Language Recognition Workshop*, pages 195–202, 2018.

[185] Zhao Yi, Wen-Chin Huang, Xiaohai Tian, Junichi Yamagishi, Rohan Kumar Das, Tomi Kinnunen, Zhen-Hua Ling, and Tomoki Toda. Voice Conversion Challenge 2020 – Intra-lingual semi-parallel and cross-lingual voice conversion –. In *Proc. Joint Workshop for the Blizzard Challenge and Voice Conversion Challenge 2020*, pages 80–98, 2020. 26

[186] Renato De Mori, Frédéric Bechet, Dilek Hakkani-Tur, Michael McTear, Giuseppe Riccardi, and Gokhan Tur. Spoken language understanding. *IEEE Signal Processing Magazine*, 25(3):50–58, 2008. 26

[187] Blaise Thomson and Steve Young. Bayesian update of dialogue state: A pomdp framework for spoken dialogue systems. *Computer Speech & Language*, 24(4):562–588, 2010. 66, 81

[188] Oliver Lemon and Olivier Pietquin. *Data-driven methods for adaptive spoken dialogue systems: Computational learning for conversational interfaces*. Springer New York, 2012. 26

[189] Tsung-Hsien Wen, Milica Gašic, Nikola Mrkšic, Lina M Rojas-Barahona, Pei-Hao Su, David Vandyke, and Steve Young. Toward multi-domain language generation using recurrent neural networks. In *NIPS Workshop on Machine Learning for Spoken Language Understanding and Interaction*, 2015. 26, 106

[190] Pei-Hao Su, Milica Gašić, Nikola Mrkšić, Lina M. Rojas-Barahona, Stefan Ultes, David Vandyke, Tsung-Hsien Wen, and Steve Young. On-line active reward learning for policy optimisation in spoken dialogue systems. In *Proceedings of Proceedings of ACL*, pages 2431–2441, 2016. 26

[191] Stefan Ultes, Lina M. Rojas-Barahona, Pei-Hao Su, David Vandyke, Dongho Kim, Iñigo Casanueva, Paweł Budzianowski, Nikola Mrkšić, Tsung-Hsien Wen, Milica Gašić, and Steve Young. PyDial: A multi-domain statistical dialogue system toolkit. In *Proceedings of ACL*, pages 73–78, 2017. 26, 84

[192] Kaisheng Yao, Baolin Peng, Geoffrey Zweig, Dong Yu, Xiaolong Li, and Feng Gao. Recurrent conditional random field for language understanding. In *Proceedings of ICASSP*, pages 4077–4081. IEEE, 2014. 29

[193] Dilek Hakkani-Tür, Gökhan Tür, Asli Celikyilmaz, Yun-Nung Chen, Jianfeng Gao, Li Deng, and Ye-Yi Wang. Multi-domain joint semantic frame parsing using bi-directional rnn-lstm. In *INTERSPEECH*, pages 715–719, 2016. 29

[194] Steve Young. CUED standard dialogue acts. *Report, Cambridge University Engineering Department, 14th October*, 2007, 2007. 29, 43, 185

[195] Matthew Henderson, Milica Gašić, Blaise Thomson, Pirros Tsiakoulis, Kai Yu, and Steve Young. Discriminative spoken language understanding using word confusion networks. In *2012 IEEE Spoken Language Technology Workshop (SLT)*, pages 176–181. IEEE, 2012. 30, 43

[196] Zhuoran Wang and Oliver Lemon. A simple and generic belief tracking mechanism for the dialog state tracking challenge: On the believability of observed information. In *Proceedings of SIGDIAL*, pages 423–432, 2013.

[197] Kai Yu, Kai Sun, Lu Chen, and Su Zhu. Constrained markov bayesian polynomial for efficient dialogue state tracking. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 23(12):2177–2188, 2015.

[198] Youngsoo Jang, Jiyeon Ham, Byung-Jun Lee, and Kee-Eung Kim. Cross-language neural dialog state tracker for large ontologies using hierarchical attention. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 26(11):2072–2082, 2018. 30

[199] John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu. From paraphrase database to compositional paraphrase model and back. *Transactions of the Association for Computational Linguistics*, 3: 345–358, 2015. 30

[200] Iulian Vlad Serban, Ryan Lowe, Peter Henderson, Laurent Charlin, and Joelle Pineau. A survey of available corpora for building data-driven dialogue systems: The journal version. *Dialogue & Discourse*, 9(1):1–49, 2018. 31

[201] Antoine Bordes, Y-Lan Boureau, and Jason Weston. Learning end-to-end goal-oriented dialog. In *Proceedings of ICLR*, 2017. 31, 80, 153

[202] Alan W Black, Susanne Burger, Alistair Conkie, Helen Hastie, Simon Keizer, Oliver Lemon, Nicolas Merigaud, Gabriel Parent, Gabriel Schubiner, Blaise Thomson, et al. Spoken dialog challenge 2010: Comparison of live and control test results. In *Proceedings of SIGDIAL*, pages 2–7, 2011. 31

[203] Antoine Raux, Brian Langner, Dan Bohus, Alan W Black, and Maxine Eskenazi. Let's go public! taking a spoken dialog system to the real world. In *Ninth EUROSPEECH*, 2005. 31

[204] Christina Bennett and Alexander I Rudnicky. The carnegie mellon communicator corpus. In *Seventh International Conference on Spoken Language Processing*, pages 341–344, 2002. 32

[205] Nicolas Schrading, Cecilia Ovesdotter Alm, Raymond Ptucha, and Christopher Homan. An analysis of domestic abuse discourse on reddit. In *Proceedings of EMNLP*, pages 2577–2583, 2015. 32

[206] Oriol Vinyals and Quoc V. Le. A neural conversational model. In *ICML Deep Learning Workshop*, 2015. 32, 80

[207] Chia-Wei Liu, Ryan Lowe, Iulian Serban, Mike Noseworthy, Laurent Charlin, and Joelle Pineau. How NOT to evaluate your dialogue system: An empirical study of unsupervised evaluation metrics for dialogue response generation. In *Proceedings of EMNLP*, pages 2122–2132, 2016. 32, 80, 152

[208] John F Kelley. An iterative design methodology for user-friendly natural language office information applications. *ACM Transactions on Information Systems (TOIS)*, 2(1):26–41, 1984. 33, 36

[209] Nikola Mrkšić. *Data-Driven Language Understanding for Spoken Dialogue Systems*. PhD thesis, University of Cambridge, 2018. 34

[210] Rohit Mishra, Elizabeth Shriberg, Sandra Upson, Joyce Chen, Fuliang Weng, Stanley Peters, Lawrence Cavedon, John Niekrasz, Hua Cheng, and Harry Bratt. A wizard of oz framework for collecting spoken human-computer dialogs. In *Eighth International Conference on Spoken Language Processing*, 2004. 36

[211] Norman M Fraser and G Nigel Gilbert. Simulating speech systems. *Computer Speech & Language*, 5 (1):81–99, 1991. 36

[212] Joseph L Fleiss. Measuring nominal scale agreement among many raters. *Psychological bulletin*, 76(5): 378, 1971. 38

[213] Michael I. Jordan. Chapter 25 - serial order: A parallel distributed processing approach. In *Neural-Network Models of Cognition*, volume 121 of *Advances in Psychology*, pages 471 – 495. North-Holland, 1997. 45

[214] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Proceedings of NIPS*, pages 5998–6008, 2017. 48, 51, 52

[215] Luheng He, Kenton Lee, Mike Lewis, and Luke Zettlemoyer. Deep semantic role labeling: What works and what's next. In *Proceedings of ACL*, pages 473–483, 2017. 48

[216] Osman Ramadan, Paweł Budzianowski, and Milica Gašić. Large-scale multi-domain belief tracking with knowledge sharing. In *Proceedings of ACL*, pages 432–437, 2018. 52

[217] AI4Bharat-IndicNLP corpus: Monolingual corpora and word embeddings for indic languages. 53, 54

[218] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. In *Proceedings of ICLR*, 2013. 53, 106

[219] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *Proceedings of EMNLP*, pages 1532–1543, 2014. 53

[220] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146, 2017. 54, 57

[221] Edouard Grave, Piotr Bojanowski, Prakhar Gupta, Armand Joulin, and Tomas Mikolov. Learning word vectors for 157 languages. In *Proceedings of LREC*, Miyazaki, Japan, May 2018. European Language Resources Association (ELRA). 54

[222] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: A system for large-scale machine learning. In *12th {USENIX} symposium on ({OSDI} 16)*, pages 265–283, 2016. 55

[223] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017. 55

[224] Michael F. McTear. *Spoken Dialogue Technology*. Springer London, 2004. 61

[225] Michael H Cohen, Michael Harris Cohen, James P Giangola, and Jennifer Balogh. *Voice user interface design*. Addison-Wesley Professional, 2004. 61

[226] Scott McGlashan, Daniel Burnett, Jerry Carter, Peter Danielsen, Jim Ferrans, Andrew Hunt, Bruce Lucas, Brad Porter, Ken Rehor, and Steph Tryphonas. Voice extensible markup language (voicexml) version 2.0. *W3C, Technical specification*, 2004. 61

[227] Dan Bohus and Alexander I Rudnicky. Ravenclaw: Dialog management using hierarchical task decomposition and an expectation agenda. In *Eighth EUROSPEECH*, 2003. 61

[228] Dan Bohus and Alexander Rudnicky. Error handling in the ravenclaw dialog management architecture. In *Proceedings of HLT & EMNLP*, pages 225–232, 2005. 61

[229] Gellért Weisz, Paweł Budzianowski, Pei-Hao Su, and Milica Gašić. 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, 2018. 62

[230] Heriberto Cuayáhuitl, Simon Keizer, and Oliver Lemon. Strategic dialogue management via deep reinforcement learning. In *Proceedings of NIPS Workshop Deep Reinforcement Learning*, 2015. 62

[231] Ziyu Wang, Victor Bapst, Nicolas Heess, Volodymyr Mnih, Remi Munos, Koray Kavukcuoglu, and Nando de Freitas. Sample efficient actor-critic with experience replay. In *Proceedings of ICLR*, 2017. 62

[232] Jost Schatzmann. *Statistical User and Error Modelling for Spoken Dialogue Systems*. PhD thesis, University of Cambridge, 2008. 62, 78, 79

[233] Rémi Munos, Tom Stepleton, Anna Harutyunyan, and Marc G Bellemare. Safe and efficient off-policy reinforcement learning. In *Proceedings of NIPS*, pages 1046–1054, 2016. 62, 75

[234] Srinivasan Janarthanam, Helen Hastie, Oliver Lemon, and Xingkun Liu. "the day after the day after tomorrow?" a machine learning approach to adaptive temporal expression generation: training and evaluation with real users. In *Proceedings of SIGDIAL*, pages 142–151, 2011. 63

[235] Sungjin Lee and Maxine Eskenazi. Pomdp-based let's go system for spoken dialog challenge. In *2012 IEEE Spoken Language Technology Workshop (SLT)*, pages 61–66. IEEE, 2012. 63

[236] George E Monahan. State of the art—a survey of partially observable markov decision processes: theory, models, and algorithms. *Management science*, 28(1):1–16, 1982. 63

[237] Leslie Pack Kaelbling, Michael L Littman, and Anthony R Cassandra. Planning and acting in partially observable stochastic domains. *Artificial intelligence*, 101(1-2):99–134, 1998. 63, 68

[238] Martin L Puterman. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014. 63

[239] Leslie Pack Kaelbling, Michael L Littman, and Andrew W Moore. Reinforcement learning: A survey. *Journal of artificial intelligence research*, 4:237–285, 1996. 63

[240] Richard Bellman. The theory of dynamic programming. Technical report, DTIC Document, 1954. 68, 85

[241] Lucian Buşoniu, Damien Ernst, Bart De Schutter, and Robert Babuška. Approximate reinforcement learning: An overview. In *2011 IEEE symposium on ADPRL*, pages 1–8. IEEE, 2011. 69

[242] Joelle Pineau, Geoff Gordon, Sebastian Thrun, et al. Point-based value iteration: An anytime algorithm for pomdps. In *IJCAI*, volume 3, pages 1025–1032, 2003. 69

[243] Guy Shani, Joelle Pineau, and Robert Kaplow. A survey of point-based pomdp solvers. *Autonomous Agents and Multi-Agent Systems*, 27(1):1–51, 2013. 69

[244] Marc Peter Deisenroth, Gerhard Neumann, Jan Peters, et al. A survey on policy search for robotics. *Foundations and trends in Robotics®*, 2(1-2):388–403, 2013. 69

[245] Reuven Y Rubinstein and Dirk P Kroese. *The cross-entropy method: a unified approach to combinatorial optimization, Monte-Carlo simulation and machine learning*. Springer Science & Business Media, 2013. 70

[246] Daan Wierstra, Tom Schaul, Tobias Glasmachers, Yi Sun, Jan Peters, and Jürgen Schmidhuber. Natural evolution strategies. *The Journal of Machine Learning Research*, 15(1):949–980, 2014. 70

[247] Tim Salimans, Jonathan Ho, Xi Chen, Szymon Sidor, and Ilya Sutskever. Evolution strategies as a scalable alternative to reinforcement learning. *arXiv preprint arXiv:1703.03864*, 2017. 70

[248] Richard S Sutton, David A McAllester, Satinder P Singh, Yishay Mansour, et al. Policy gradient methods for reinforcement learning with function approximation. In *Proceedings of NIPS*, volume 12, pages 1057–1063, 1999. 70, 71, 72, 74

[249] Peter W Glynn. Likelihood ratio gradient estimation for stochastic systems. *Communications of the ACM*, 33(10):75–84, 1990. 70, 192

[250] Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3):229–256, 1992. 71, 73

[251] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018. 72, 73, 74

[252] Evan Greensmith, Peter L Bartlett, and Jonathan Baxter. Variance reduction techniques for gradient estimates in reinforcement learning. *Journal of Machine Learning Research*, 5:1471–1530, 2004. 72

[253] Ivo Grondman, Lucian Busoniu, Gabriel AD Lopes, and Robert Babuska. A survey of actor-critic reinforcement learning: Standard and natural policy gradients. *IEEE Transactions on Systems, Man, and Cybernetics*, 42(6):1291–1307, 2012. 72

[254] Hamid Reza Maei, Csaba Szepesvári, Shalabh Bhatnagar, and Richard S Sutton. Toward off-policy learning control with function approximation. In *Proceedings of ICML*, 2010. 73

[255] Long-Ji Lin. Self-improving reactive agents based on reinforcement learning, planning and teaching. *Machine learning*, 8(3-4):293–321, 1992. 75, 86

[256] Nicolas Meuleau, Leonid Peshkin, Leslie P Kaelbling, and Kee-Eung Kim. Off-policy policy search. *MIT Articical Intelligence Laboratory*, 2000. 75, 86

[257] John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. In *Proceedings of ICLR*, 2016. 75

[258] Doina Precup, Richard S Sutton, and Sanjoy Dasgupta. Off-policy temporal-difference learning with function approximation. In *ICML*, pages 417–424, 2001. 76

[259] Jost Schatzmann and Steve Young. The hidden agenda user simulation model. *IEEE transactions on audio, speech, and language processing*, 17(4):733–747, 2009. 77, 78, 87

[260] Lihong Li, Jason D Williams, and Suhrid Balakrishnan. Reinforcement learning for dialog management using least-squares policy iteration and fast feature selection. In *INTERSPEECH*, 2009. 77

[261] Yasuhiro Minami, Ryuichiro Higashinaka, Kohji Dohsaka, Toyomi Meguro, and Eisaku Maeda. Trigram dialogue control using pomdps. In *Proceedings of SLT*, pages 336–341. IEEE, 2010. 77

[262] Jost Schatzmann, Karl Weilhammer, Matt Stuttle, and Steve Young. A survey of statistical user simulation techniques for reinforcement-learning of dialogue management strategies. *The knowledge engineering review*, 21(2):97–126, 2006. 77

[263] Konrad Scheffler and Steve Young. Probabilistic simulation of human-machine dialogues. In *Proceedings of ICASSP*. IEEE, 2000. 78

[264] Olivier Pietquin and Thierry Dutoit. A probabilistic framework for dialog simulation and optimal strategy learning. *IEEE Transactions on Audio, Speech, and Language Processing*, 14(2):589–599, 2006. 78

[265] Verena Rieser and Oliver Lemon. Cluster-based user simulations for learning dialogue strategies. In *INTERSPEECH*, 2006.

[266] Kallirroi Georgila, James Henderson, and Oliver Lemon. User simulation for spoken dialogue systems: Learning and evaluation. In *INTERSPEECH*, 2006. 78

[267] Sangkeun Jung, Cheongjae Lee, Kyungduk Kim, Minwoo Jeong, and Gary Geunbae Lee. Data-driven user simulation for automated evaluation of spoken dialog systems. *Computer Speech & Language*, 23 (4):479–509, 2009. 78

[268] Jost Schatzmann, Blaise Thomson, Karl Weilhammer, Hui Ye, and Steve Young. Agenda-based user simulation for bootstrapping a POMDP dialogue system. In *Human Language Technologies 2007: NAACL*, pages 149–152, 2007. 79

[269] Blaise Thomson. *Statistical methods for spoken dialogue management*. PhD thesis, University of Cambridge, 2009. 79

[270] Jost Schatzmann, Blaise Thomson, and Steve Young. Error simulation for training statistical dialogue systems. In *IEEE Workshop on Automatic Speech Recognition & Understanding (ASRU)*, pages 526–531. IEEE, 2007. 80

[271] Lynette Hirschman and Henry S Thompson. Overview of evaluation in speech and natural language processing. *New York, NY, USA: Cambridge University Press*, pages 409–414, 1997. 80

[272] Jade Goldstein, Mark Kantrowitz, Vibhu Mittal, and Jaime Carbonell. Summarizing text documents: Sentence selection and evaluation metrics. In *Proceedings of SIGIR*, pages 121–128, 1999. 80

[273] Oliver Lemon, Kallirroi Georgila, and James Henderson. Evaluating effectiveness and portability of reinforcement learned dialogue strategies with real users: the talk towninfo evaluation. In *Proceedings of SLT*, pages 178–181. IEEE, 2006. 80, 81

[274] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of ACL*, pages 311–318, 2002. 80, 106

[275] Satanjeev Banerjee and Alon Lavie. METEOR: An automatic metric for mt evaluation with improved correlation with human judgments. In *Proceedings of the ACL workshop*, pages 65–72, 2005. 80

[276] Daniel Dewey. Reinforcement learning and the reward engineering principle. In *Proceedings of AAAI*, 2014. 81

[277] Iñigo Casanueva, Thomas Hain, Heidi Christensen, Ricard Marxer, and Phil Green. Knowledge transfer between speakers for personalised dialogue management. In *Proceedings of SIGDIAL*, pages 12–21, 2015. 81

[278] Milica Gašić. *Statistical Dialogue Modelling*. PhD thesis, University of Cambridge, 2011. 83

[279] Todd Hester, Matej Vecerik, Olivier Pietquin, Marc Lanctot, Tom Schaul, Bilal Piot, Dan Horgan, John Quan, Andrew Sendonaris, Ian Osband, et al. Deep Q-Learning from demonstrations. In *Proceedings of AAAI*, 2018. 83

[280] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016. 83, 85

[281] Yaakov Engel. *Algorithms and Representations for Reinforcement Learning*. PhD thesis, Hebrew University of Jerusalem Jerusalem, 2005. 85

[282] Shixiang Gu, Timothy Lillicrap, Zoubin Ghahramani, Richard E Turner, and Sergey Levine. Q-prop: Sample-efficient policy gradient with an off-policy critic. In *Proceedings of ICLR*, 2016. 85

[283] Blaise Thomson. *Statistical methods for spoken dialogue management*. Springer Science & Business Media, 2013. 86

[284] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 87

[285] Zhuoran Wang, Tsung-Hsien Wen, Pei-Hao Su, and Yannis Stylianou. Learning domain-independent dialogue policies via ontology parameterisation. In *Proceedings of SIGDIAL*, pages 412–416, 2015. 89

[286] Alexandros Papangelis and Yannis Stylianou. Single-model multi-domain dialogue management with deep learning. In *Advanced Social Interaction with Agents*, pages 71–77. Springer, 2019. 89

[287] Filip Jurčíček, Simon Keizer, Milica Gašić, Francois Mairesse, Blaise Thomson, Kai Yu, and Steve Young. Real user evaluation of spoken dialogue systems using amazon mechanical turk. In *Proceedings of INTERSPEECH*, volume 11, 2011. 93

[288] Lu Chen, Zhi Chen, Bowen Tan, Sishan Long, Milica Gasic, and Kai Yu. Agentgraph: Towards universal dialogue management with structured deep reinforcement learning. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 2019. 94

[289] Filip Jurčíček, Ondřej Dušek, Ondřej Plátek, and Lukáš Žilka. Alex: A statistical dialogue systems framework. In *International Conference on Text, Speech, and Dialogue*, pages 587–594. Springer, 2014. 96

[290] Zhichao Hu, Gabrielle Halberg, Carolynn R Jimenez, and Marilyn A Walker. Entrainment in pedestrian direction giving: How many kinds of entrainment? In *Situated Dialog in Speech-Based Human-Computer Interaction*, pages 151–164. Springer, 2016. 96

[291] Omkar Dhariya, Shrikant Malviya, and Uma Shanker Tiwary. A hybrid approach for hindi-english machine translation. In *2017 International Conference on Information Networking (ICOIN)*, pages 389–394. IEEE, 2017. 98

[292] Klaus Greff, Rupesh K Srivastava, Jan Koutník, Bas R Steunebrink, and Jürgen Schmidhuber. Lstm: A search space odyssey. *IEEE transactions on neural networks and learning systems*, 28(10):2222–2232, 2016. 99

[293] Anoop Kunchukuttan, Pratik Mehta, and Pushpak Bhattacharyya. The IIT Bombay English-Hindi Parallel Corpus. *Language Resources and Evaluation Conference*, 2018. 106

[294] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*, 2016. 108

[295] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The elements of statistical learning: data mining, inference, and prediction*. Springer Science & Business Media, 2009. 115, 128, 129

[296] A. W. Black. CMU-Indic speech synthesis databases. URL http://festvox.org/cmu_indic/index.html. accessed 2018-12-15. 116

[297] Arun Baby, NL Nishanthi, Anju Leela Thomas, and Hema A Murthy. Resources for indian languages. In *International Conference on Text, Speech, and Dialogue*, pages 514–521. Springer, 2016. 116

[298] Alok Parlikar, Sunayana Sitaram, Andrew Wilkinson, and Alan W Black. The festvox indic frontend for grapheme to phoneme conversion. In *WILDRE: Workshop on Indian Language Data-Resources and Evaluation*, 2016. 116

[299] Shrikant Malviya, Rohit Mishra, and Uma Shanker Tiwary. Structural analysis of hindi phonetics and a method for extraction of phonetically rich sentences from a very large hindi text corpus. In *2016 Conference of O-COCOSDA*, pages 188–193. IEEE, 2016. 116

[300] Anusha Prakash, Jeena J Prakash, and Hema A Murthy. Acoustic analysis of syllables across indian languages. In *INTERSPEECH*, pages 327–331, 2016. 116

[301] Ingmar Steiner and Sébastien Le Maguer. Creating new language and voice components for the updated MaryTTS text-to-speech synthesis platform. In *Proceedings of LREC*. European Language Resources Association (ELRA), 2018. 118

[302] David T. Chappell and John H.L. Hansen. A comparison of spectral smoothing methods for segment concatenation based speech synthesis. *Speech Communication*, 36(3):343 – 373, 2002. ISSN 0167-6393. 119

[303] A. W. Black. Perfect synthesis for all of the people all of the time. In *Proceedings of 2002 IEEE Workshop on Speech Synthesis, 2002.*, pages 167–170, Sept 2002. 119

[304] K. Prahallad, A. W. Black, and R. Mosur. Sub-phonetic modeling for capturing pronunciation variations for conversational speech synthesis. In *Proceedings of ICASSP*, volume 1, pages 853–856. IEEE, 2006. 121

[305] Satoshi Imai. Cepstral analysis synthesis on the mel frequency scale. In *Proceedings of ICASSP*, volume 8, pages 93–96. IEEE, 1983. 121

[306] Toshiaki Fukada, Keiichi Tokuda, Takao Kobayashi, and Satoshi Imai. An adaptive algorithm for mel-cepstral analysis of speech. In *Proceedings of ICASSP*, volume 1, pages 137–140. IEEE, 1992. 122

[307] Yu-Yun Chang. Evaluation of tts systems in intelligibility and comprehension tasks. In *proceedings of the 23rd Conference on Computational Linguistics and Speech Processing*, pages 64–78, 2011. 123

[308] HA Sydeserff, RJ Caley, Stephen D Isard, Mervyn A Jack, Alex IC Monaghan, and Jo Verhoeven. Evaluation of speech synthesis techniques in a comprehension task. *Speech communication*, 11(2-3):189–194, 1992. 123

[309] Wayland M Parrish. The concept of "naturalness". *Quarterly Journal of Speech*, 37(4):448–454, 1951. 123

[310] Carlos Monzo, Ignasi Iriondo, and Joan Claudi Socoró. Voice quality modelling for expressive speech synthesis. *The Scientific World Journal*, 2014, 2014. 123

[311] James R Lewis. Effect of speaker and sampling rate on mos-x ratings of concatenative tts voices. In *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, volume 48, pages 759–763. SAGE Publications Sage CA: Los Angeles, CA, 2004. 123

[312] Philipos C Loizou. *Speech enhancement: theory and practice*. CRC press, 2013. 124

[313] Philipos C Loizou. Speech quality assessment. In *Multimedia analysis, processing and communications*, pages 623–654. Springer, 2011. 124, 134

[314] Pavlos Papadopoulos, Ruchir Travadi, and Shrikanth Narayanan. Global snr estimation of speech signals for unknown noise conditions using noise adapted non-linear regression. *INTERSPEECH*, pages 3842–3846, 2017. 124

[315] Bo Wei and Jerry D Gibson. Comparison of distance measures in discrete spectral modeling. Master's thesis, S. M. U., 2001. 124

[316] Brian CJ Moore. *An introduction to the psychology of hearing*. Brill, 2012. 124

[317] Bob Novorita. Incorporation of temporal masking effects into bark spectral distortion measure. In *Proceedings of ICASSP*, volume 2, pages 665–668. IEEE, 1999. 124

[318] Arun Baby, NL Nishanthi, Anju Leela Thomas, and Hema A Murthy. A unified parser for developing indian language text to speech synthesizers. In *International Conference on Text, Speech, and Dialogue*, pages 514–521. Springer, 2016. 125

[319] Florian Eyben, Felix Weninger, Florian Gross, and Björn Schuller. Recent developments in opensmile, the munich open-source multimedia feature extractor. In *Proceedings of the 21st ACM international conference on Multimedia*, pages 835–838. ACM, 2013. 125

[320] Björn Schuller. *Automatische Emotionserkennung aus sprachlicher und manueller Interaktion*. PhD thesis, Technische Universität München, 2006. 126

[321] Björn W Schuller, Stefan Steidl, Anton Batliner, Julia Hirschberg, Judee K Burgoon, Alice Baird, Aaron C Elkins, Yue Zhang, Eduardo Coutinho, and Keelan Evanini. The interspeech 2016 computational paralinguistics challenge: Deception, sincerity & native language. In *INTERSPEECH*, volume 2016, pages 2001–2005, 2016. 127, 134, 142

[322] Roman Rosipal and Nicole Krämer. Overview and recent advances in partial least squares. In *International Statistical and Optimization Perspectives Workshop" Subspace, Latent Structure and Feature Selection"*, pages 34–51. Springer, 2005. 129

[323] Chih-Chung Chang and Chih-Jen Lin. Libsvm: A library for support vector machines. *ACM transactions on intelligent systems and technology (TIST)*, 2(3):1–27, 2011. 129

[324] Björn Schuller, Stefan Steidl, and Anton Batliner. The interspeech 2009 emotion challenge. In *INTERSPEECH*, 2009. 134, 142

[325] Björn Schuller, Stefan Steidl, Anton Batliner, Felix Burkhardt, Laurence Devillers, Christian Müller, and Shrikanth S Narayanan. The interspeech 2010 paralinguistic challenge. In *INTERSPEECH*, 2010. 134, 142

[326] Björn Schuller, Stefan Steidl, Anton Batliner, Florian Schiel, and Jarek Krajewski. The interspeech 2011 speaker state challenge. In *INTERSPEECH*, 2011. 134, 142

[327] Björn Schuller, Stefan Steidl, Anton Batliner, Elmar Nöth, Alessandro Vinciarelli, Felix Burkhardt, Rob van Son, Felix Weninger, Florian Eyben, Tobias Bocklet, et al. The interspeech 2012 speaker trait challenge. In *INTERSPEECH*, 2012. 134, 142

[328] Björn Schuller, Stefan Steidl, Anton Batliner, Alessandro Vinciarelli, Klaus Scherer, Fabien Ringeval, Mohamed Chetouani, Felix Weninger, Florian Eyben, Erik Marchi, et al. The interspeech 2013 computational paralinguistics challenge: social signals, conflict, emotion, autism. In *INTERSPEECH*, 2013. 134, 142

[329] Michel Valstar, Björn Schuller, Kirsty Smith, Florian Eyben, Bihan Jiang, Sanjay Bilakhia, Sebastian Schnieder, Roddy Cowie, and Maja Pantic. Avec 2013: the continuous audio/visual emotion and depression recognition challenge. In *Proceedings of the 3rd ACM international workshop on Audio/visual emotion challenge*, pages 3–10. ACM, 2013. 134, 142

[330] Florian Eyben, Klaus R Scherer, Björn W Schuller, Johan Sundberg, Elisabeth André, Carlos Busso, Laurence Y Devillers, Julien Epps, Petri Laukka, Shrikanth S Narayanan, et al. The geneva minimalistic acoustic parameter set (gemaps) for voice research and affective computing. *IEEE Transactions on Affective Computing*, 7(2):190–202, 2016. 134, 142

[331] Ezequiel Uriel. Hypothesis testing in the multiple regression model. *Universidad de Valencia: Department of economics*, 2013. 138

[332] R Thangarajan and AM Natarajan. Syllable based continuous speech recognition for tamil. *South Asian language review*, 18(1):72–85, 2008. 139

[333] Carlos Toshinori Ishi, Jun Arai, and Norihiro Hagita. Prosodic analysis of attention-drawing speech. In *INTERSPEECH*, pages 909–913, 2017. 142

[334] Florian Eyben. *Real-time Speech and Music Classification by Large Audio Feature Space Extraction*. Springer Publishing Company, Incorporated, 1st edition, 2016. 142

[335] Björn W Schuller. *Intelligent audio analysis*. Springer, 2013. 143

[336] Christoph R Norrenbrock, Florian Hinterleitner, Ulrich Heute, and Sebastian Moller. Instrumental assessment of prosodic quality for text-to-speech signals. *IEEE Signal Processing Letters*, 19(5):255–258, 2012. 143

[337] T Falk and W Chan. Single ended method for objective speech quality assessment in narrowband telephony applications. *ITU-T*, page 563, 2004. 144

[338] Volodya Grancharov, David Yuheng Zhao, Jonas Lindblom, and W Bastiaan Kleijn. Low-complexity, nonintrusive speech quality assessment. *IEEE Transactions on Audio, Speech, and Language Processing*, 14(6):1948–1956, 2006. 144

[339] Doh-Suk Kim. Anique: An auditory model for single-ended speech quality estimation. *IEEE Transactions on Speech and Audio Processing*, 13(5):821–831, 2005. 144

[340] Ryan Lowe, Michael Noseworthy, Iulian V Serban, Nicolas Angelard-Gontier, Yoshua Bengio, and Joelle Pineau. Towards an automatic turing test: Learning to evaluate dialogue responses. In *Proceedings of ACL*, 2017. 152

[341] Andreas Stolcke, Klaus Ries, Noah Coccaro, Elizabeth Shriberg, Rebecca Bates, Daniel Jurafsky, Paul Taylor, Rachel Martin, Carol Van Ess-Dykema, and Marie Meteer. Dialogue act modeling for automatic tagging and recognition of conversational speech. *Computational linguistics*, 26(3):339–373, 2000. 152

[342] Tsung-Hsien Wen, Yishu Miao, Phil Blunsom, and Steve Young. Latent intention dialogue models. In *Proceedings of ICML*, pages 3732–3741, 2017. 152

[343] Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. Pointer networks. In *Proceedings of NIPS*, 2015. 152

[344] Xuesong Yang, Yun-Nung Chen, Dilek Hakkani-Tür, Paul Crook, Xiujun Li, Jianfeng Gao, and Li Deng. End-to-end joint learning of natural language understanding and dialogue manager. In *Proceedings of ICASSP*, pages 5690–5694. IEEE, 2017. 153

[345] Susan E Brennan. Conversation and dialogue. In *H. Pashler (Ed.) Encyclopedia of the Mind*. Sage Publications, 2012. 153

[346] Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C Lawrence Zitnick, and Devi Parikh. Vqa: Visual question answering. In *Proceedings of the IEEE international conference on computer vision*, pages 2425–2433, 2015. 153

[347] Peng Zhang, Yash Goyal, Douglas Summers-Stay, Dhruv Batra, and Devi Parikh. Yin and yang: Balancing and answering binary visual questions. In *Proceedings of the IEEE Conference on CVPR*, pages 5014–5022, 2016. 153

[348] Yash Goyal, Tejas Khot, Douglas Summers-Stay, Dhruv Batra, and Devi Parikh. Making the v in vqa matter: Elevating the role of image understanding in visual question answering. In *Proceedings of the IEEE Conference on CVPR*, pages 6904–6913, 2017.

[349] Abhishek Das, Satwik Kottur, Khushi Gupta, Avi Singh, Deshraj Yadav, José MF Moura, Devi Parikh, and Dhruv Batra. Visual dialog. In *Proceedings of the IEEE Conference on CVPR*, pages 326–335, 2017.

[350] Abhishek Das, Satwik Kottur, José MF Moura, Stefan Lee, and Dhruv Batra. Learning cooperative visual dialog agents with deep reinforcement learning. In *Proceedings of the IEEE international conference on computer vision*, pages 2951–2960, 2017.

[351] Harm De Vries, Florian Strub, Sarath Chandar, Olivier Pietquin, Hugo Larochelle, and Aaron Courville. Guesswhat?! visual object discovery through multi-modal dialogue. In *Proceedings of the IEEE Conference on CVPR*, pages 5503–5512, 2017.

[352] Satwik Kottur, José M. F. Moura, Devi Parikh, Dhruv Batra, and Marcus Rohrbach. CLEVR-dialog: A diagnostic dataset for multi-round reasoning in visual dialog. In *Proceedings of NAACL*, pages 582–595, 2019. 153

Appendix A

Allahabad Restaurant Domain

Task-oriented dialogue systems generally use the *slot-filling* mechanism to proceed a dialogue. The slots of a slot-based dialogue system specify its domain, i.e. the scope system can talk about and the tasks it can help users to accomplish. The slots also define the set of possible actions the system can take, the possible semantics of the user utterances (see Appendix B) and the possible dialogue states (see Section 3.4).

In the information-seeking dialogue scenario, where the goal of the dialogue system is to allow the user to search a database for data items by specifying constraints, the slots are the attributes of the entities in the database, and a set of slot-value pairs form a search query in this case.

The set of all slots S is composed of two subsets: the informable slots S_{inf}, and the requestable slots S_{req}, such that $S = S_{inf} \cup S_{req}$. Informable slots are attributes of the entities in the database that the user may use to constrain their search. On the other hand, requestable slots are attributes that users may ask the value of, but may not necessarily be allowed to specify a value as a constraint. A typical example of a requestable slot that is not informable is the phone number, which the user may ask for but would not give as a constraint ("मैं एक रेस्तरां खोज रहा हूँ जिसका फोन नंबर `0532355166` है।") but the user may ask the value for ("क्या मुझे उस रेस्तरां का फोन नंबर मिल सकता है? "). In addition, these two sets of slots are not necessarily disjoint. Requestable slots are typically not informable, while informable slots are typically requestable.

The domain used for evaluations in this thesis is *restaurant information*. The ontology and database for the domain are collected and experimented about finding a restaurant in the Allahabad area. A summary of the domain is presented in Table A.1 & Table A.2. Table A.1 signifies the distribution of venues (restaurants) based on the domain slots, i.e. price range and area. On the other hand, Table A.2 shows the slot-specific details categorised into S_{inf} (informable) and S_{req} (requestable) types.

Table A.1 Allahabad restaurant database distribution of venues based on price range and area.

Sno	Price range	Area		Number of restaurants
1.	महंगा (High)	उत्तर	(North)	6
		केंद्र	(Center)	41
		दक्षिण	(South)	5
		पश्चिम	(West)	7
		पूर्व	(East)	5
2.	मध्यम (Medium)	उत्तर	(North)	3
		केंद्र	(Center)	21
		दक्षिण	(South)	2
		पश्चिम	(West)	3
		पूर्व	(East)	3
3	सस्ता (Low)	उत्तर	(North)	2
		केंद्र	(Center)	15
		दक्षिण	(South)	2
		पश्चिम	(West)	2
		पूर्व	(East)	1

Table A.2 Ontology (slots) used in the Allahabad Restaurant search domain. All informable slots are also requestable. The group $S_{req} \setminus S_{inf}$ displays the requestable slots that are not informable.

Sno	Type	Slot	Total Number of Values
1.	S_{inf}	food	34
2.		area	5
3.		price range	3
1.	$S_{req} \setminus S_{inf}$	address	-
2.		phone	-
3.		postcode	-
4.		name	-

Appendix B

Dialogue Act Types

Dialogue acts offer a shallow representation of the semantics for the user's and the system's prompt. For the input side, the dialogue system understands the underlying semantics in the user's response by mapping text into one of the dialogue act taxonomies. At the output side, it transforms the input system act to a natural response, where system acts represent system actions or intentions associated with relevant slot-value information. The *CUED*[1] *dialogue act* taxonomy [194] is adopted in the entire thesis and described in this appendix. It is a relatively general format for representing the semantics of slot-based and task-oriented dialogues. A complete list of dialogue acts with their descriptions is given in Table B.1.

Consider a dialogue doma in containing a set of slots $S = S_{inf} \cup S_{req}$, where S_{inf} and S_{req} represents the set of informable and requestable slots. Let V_s denote the set of possible values for a slot $s \in S$. (The terminology and description of the domain studied in this thesis are explained in Appendix A).

A dialogue act is represented as the combination of two components: a dialogue act type, followed by a set of slot-value pairs (optional):

`DialActType(`s_1`=`v_1`, `s_2`=`v_2`,...`s_n`=`v_n`)`

The `DialActType` is the type of dialogue act, such as `inform`, `request`, or `confirm`. It is followed by slot-value pairs `s=v` identified from the utterance, where `s` or `v` can be null, e.g. `area=`पूर्व, `address=`. Consider an example of dialogue act with `DialActType=inform` and `SV={food=`राजस्थानी`,price range=`महंगा`}`. This dialogue act can be written in shorthand notation as follows: `inform(food=`राजस्थानी`,price range=`महंगा`)`. The DA corresponds to the abstract meaning of the following description: मैं एक महंगा रेस्तरां खोज रहा हूँ जहाँ राजस्थानी खाना मिलता हो।.

This thesis focusses on developing a spoken dialogue system in Hindi. We emphasise the dialogue acts to represent the semantics in both the DST and NLDG components. More details are described in Table B.1, where CUED dialogue acts definitions are given. First column represents the dialogue act, second and third denote

[1]CUED: A Dialogue Systems Group at Cambridge University Engineering Department (CUED)

whether the dialogue act is applicable to the system, the user or both, and in the final column, the dialogue act description is given.

Table B.1 A list of dialogue acts.

Dialogue Act	System	User	Description
hello()	✓	✓	start the dialogue
hello(a=x,b=y, . . .)	×	✓	start the dialogue with information a=x, b=y, ...
silence()	×	✓	start the dialogue with information a=x, b=y, ...
thankyou()	×	✓	implicit positive answer from the user
ack()	×	✓	back-channel e.g. uh huh, ok, etc
bye()	✓	✓	end the dialogue
hangup()	×	✓	user hangs up
inform(a=x, b=y, . . .)	✓	✓	give information a=x, b=y, ...
inform(name=none)	✓	×	inform that no matched entity is found
inform(a!=x, . . .)	×	✓	inform that a is not equal to x
inform(a=dontcare, . . .)	×	✓	the user does not care about the value of a
request(a)	✓	✓	request value of a
request(a, b=x, . . .)	✓	✓	request value of a given b=x,...
reqalts()	×	✓	request an alternative solution
reqalts(a=x, . . .)	×	✓	request an alternative solution with a=x,...
reqalts(a=dontcare, . . .)	×	✓	request an alternative solution relaxing constraint a
reqmore()	✓	×	inquire if user wants anything more
reqmore(a=dontcare)	✓	×	inquire if user would like to relax a
reqmore()	×	✓	request more information about the current solution
reqmore(a=x,b=y, . . .)	×	✓	request more information given a=x, b=y, ...
confirm(a=x,b=y, . . .)	✓	✓	confirm a=x, b=y, ...
confirm(a!=x, . . .)	✓	✓	confirm a!=x, ...
confirm(name=none)	×	✓	confirm that no suitable entity is
confirm(a=dontcare,...)	✓	✓	confirm that a is a “don’t care” value
confreq(a=x,...,c=z,d)	✓	×	confirm a=x, ... , c=z and request value of d
select(a=x, a=y)	✓	×	select either a=x or a=y
affirm()	✓	✓	simple yes response
affirm(a=x,b=y, . . .)	✓	✓	affirm and give further info a=x, b=y, ...
negate()	✓	✓	simple no response
negate(a=x)	✓	✓	negate and provide the corrected value for a
negate(a=x,b=y, . . .)	✓	✓	negate(a=x) and give further info b=y, ...
deny(a=x,b=y)	×	✓	inform that a!=x and give further info b=y, ...
repeat()	✓	✓	request to repeat last act
help()	×	✓	request for help
restart()	×	✓	request to restart
null()	✓	✓	null act, does nothing

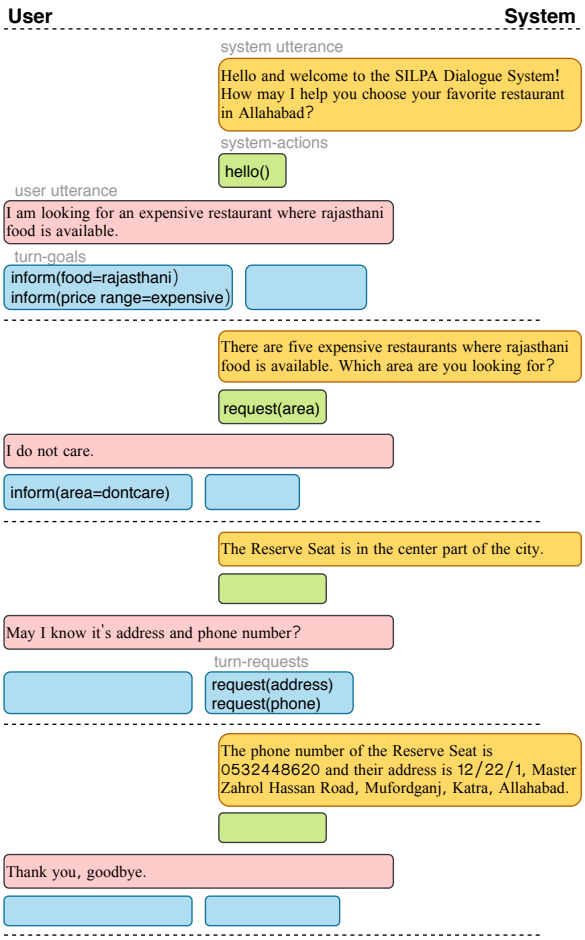
Table B.2 Examples of dialogue acts with corresponding realisations in Hindi, in the Allahabad restaurant information domain.

Dialogue Act	Utterance
User actions	
inform(area=दक्षिण, price range=मध्यम)	मुझे दक्षिण की तरफ एक मध्यम कीमत वाला रेस्टोरेंट चाहिए।
inform(area=dontcare, food=बंगाली)	मैं किसी भी क्षेत्र में बंगाली रेस्तरां की तलाश कर रहा हूं।
inform(food=कोरियन,price=सस्ता), request(phone,address)	क्या तुम मुझे एक कोरियन रेस्तरां का फ़ोन नंबर और पता दे सकते हो? ध्यान रहे मैं गरीब हूँ।
inform(area=पूर्व,food=ब्रिटिश), request(phone)	मैं शहर के पूर्व में एक ब्रिटिश रेस्टोरेंट खोज रहा हूँ। मुझे फ़ोन नंबर भी चाहिए।
request(address,phone)	क्या मुझे फ़ोन नंबर और पता मिल सकता है?
System prompts	
hello()	नमस्कार, सिल्पा डायलाग सिस्टम में आपका स्वागत है! मैं आपकी इलाहाबाद में मन चाहा रेस्टोरेंट चुननें में किस प्रकार सहायता कर सकती हूं?
reqmore()	क्या मैं आपकी कुछ और सहायता कर सकती हूँ?
inform(name=ब्लिस रेस्ट्रो, area=केंद्र,food=जैपनीज)	ब्लिस रेस्ट्रो एक जैपनीज रेस्टॉरेंट है जो कि शहर के केंद्र भाग में है।
inform(name=none, area=पश्चिम,food=मराठी)	मुझे माफ कीजिये लेकिन शहर के पश्चिम भाग में मराठी भोजन परोसने वाला कोई भी रेस्टोरेंट नहीं है।
inform(name=ओल्ड स्कूल कैफ़े, address=सिविल लाइंस, इलाहाबाद, phone=0532360966)	ओल्ड स्कूल कैफ़े का पता सिविल लाइंस, इलाहाबाद है और आप उन्हें फोन पर 0532360966 पर पहुँच सकते हैं।
request(food)	आप किस तरह का खाना पसंद करेंगे?

Appendix C

English Translation of Figure 3.1

This section presents English translation of the dialogue shown in Figure 3.1. Each turn, separated by the dashed lines, contains a *system utterance* (yellow) followed by corresponding *system-actions* (green) as well as *user utterance* (red) comes with the specified *turn-goals* and *turn-requests* (blue).



Appendix D

Proof of unbiased baseline in A2CER

Lets, recall the policy gradient expression for A2CER (from Equation 4.29):

$$\nabla_\theta J(\theta) = E_{\pi_\theta}\big[\rho \nabla_\theta \log \pi_\theta(a|\mathbf{b}) A_w(\mathbf{b}, a)\big] \tag{D.1}$$

where, ∇_θ is the gradient function, ρ denotes sampling ratio, $\pi_\theta(a|\mathbf{b})$ expresses the action probability distribution a with target policy θ over the belief distribution $\mathbf{b}$ and $A_w(\mathbf{b}, a)\big]$ is the advantage function of critic policy w.

Using the expression of advantage function $A_w(\mathbf{b}, a)$ from Equation 4.30, policy gradient function of A2CER would become:

$$\nabla_\theta J(\theta) = E_{\pi_\theta}\Big[\rho \nabla_\theta \log \pi_\theta(a|\mathbf{b})\big(r_t + \gamma V_w(\mathbf{b}_{t+1}) - V_w(\mathbf{b}_t)\big)\Big] \tag{D.2}$$

We can rearrange the expression as below:

$$\nabla_\theta J(\theta) = E_{\pi_\theta}\Big[\rho \nabla_\theta \log \pi_\theta(a|\mathbf{b}) r_t + \rho \nabla_\theta \log \pi_\theta(a|\mathbf{b}) \gamma V_w(\mathbf{b}_{t+1}) - \rho \nabla_\theta \log \pi_\theta(a|\mathbf{b}) V_w(\mathbf{b}_t)\Big] \tag{D.3}$$

The above equation is equivalent to $E(X + Y - Z)$. Due to the linearity of expectation, we can rearrange the $E(X + Y - Z)$ as $E(X) + E(Y) - E(Z)$. So the above equation is modified as below:

$$\nabla_\theta J(\theta) = E_{\pi_\theta}\Big[\rho \nabla_\theta \log \pi_\theta(a|\mathbf{b}) r_t\Big] + E_{\pi_\theta}\Big[\rho \nabla_\theta \log \pi_\theta(a|\mathbf{b}) \gamma V_w(\mathbf{b}_{t+1})\Big] - E_{\pi_\theta}\Big[\rho \nabla_\theta \log \pi_\theta(a|\mathbf{b}) V_w(\mathbf{b}_t)\Big] \tag{D.4}$$

For the generalisation, assume $(a|\mathbf{b})$ as (τ) and remove the constant terms, i.e. ρ, γ:

$$\nabla_\theta J(\theta) = E_{\pi_\theta}\Big[\nabla_\theta \log \pi_\theta(\tau) r_t\Big] + E_{\pi_\theta}\Big[\nabla_\theta \log \pi_\theta(\tau) V_w(\mathbf{b}_{t+1})\Big] - E_{\pi_\theta}\Big[\nabla_\theta \log \pi_\theta(\tau) V_w(\mathbf{b}_t)\Big] \tag{D.5}$$

Based on $E(x) = \int_x p(x)xdx$, each term in the RHS of above equation can be expanded to:

$$\nabla_\theta J(\theta) = \int_a \pi_\theta(a)\nabla_\theta \log \pi_\theta(a) r_t da \; + \int_a \pi_\theta(a)\nabla_\theta \log \pi_\theta(a) V_w(\mathbf{b}_{t+1}) da \; - \int_a \pi_\theta(a)\nabla_\theta \log \pi_\theta(a) V_w(\mathbf{b}_t) da \tag{D.6}$$

Again, using the *Likelihood ratio trick* [249]: $\nabla \log p(x) = \frac{\nabla p(x)}{p(x)} \Rightarrow p(x)\nabla \log p(x) = \nabla p(x)$, the above equation can be modified as:

$$\nabla_\theta J(\theta) = \int_a \nabla_\theta \pi_\theta(a) r_t da + \int_a \nabla_\theta \pi_\theta(a) V_w(\mathbf{b}_{t+1}) da - \int_a \nabla_\theta \pi_\theta(a) V_w(\mathbf{b}_t) da \tag{D.7}$$

$V_w(\mathbf{b}_t)$ and $V_w(\mathbf{b}_{t+1})$ are not the function of action a and ∇_θ is a linear operator, so these terms can be taken out from integral, but not the r_t as it is the function of (a_t, s_t):

$$\nabla_\theta J(\theta) = \nabla_\theta \int_a \pi_\theta(a) r_t da \; + V_w(\mathbf{b}_{t+1}) \nabla_\theta \int_a \pi_\theta(a) da \; - V_w(\mathbf{b}_t) \nabla_\theta \int_a \pi_\theta(a) da \tag{D.8}$$

As the integral of probability distribution is always 1, $\int_a \pi_\theta(a) da = 1$:

$$\nabla_\theta J(\theta) = \nabla_\theta \int_a \pi_\theta(a) r_t da \; + V_w(\mathbf{b}_{t+1}) \nabla_\theta 1 \; - V_w(\mathbf{b}_t) \nabla_\theta 1 \tag{D.9}$$

As the gradient of a constant is always zero, i.e. $\nabla_\theta 1 = 0$, second and third term in the RHS of above Equation would be zero:

$$\nabla_\theta J(\theta) = \nabla_\theta \int_a \pi_\theta(a) r_t da \tag{D.10}$$

Hence, it is proved through the derivation above that adding baseline function in A2CER has no bias on gradient estimate.

Appendix E

TTS Evaluation

E.1 Mean Opinion Score (MOS) Questionnaire Form

1. Listening Effort: Please rate the degree of effort you had to make to understand the message.

   Impossible even with much effort 1 2 3 4 5 6 7 No effort required

2. Comprehension Problems: Were single words hard to understand?

   All words hard to understand 1 2 3 4 5 6 7 All words easy to understand

3. Speech Sound Articulation: Were the speech sounds clearly distinguishable?

   Not at all clear 1 2 3 4 5 6 7 Very clear

4. Precision: Was the articulation of speech sounds precise?

   Slurred or imprecise 1 2 3 4 5 6 7 Precise

5. Voice Pleasantness: Was the voice you heard pleasant to listen to?

   Very unpleasant 1 2 3 4 5 6 7 Very pleasant

6. Voice Naturalness: Did the voice sound natural?

   Very unnatural 1 2 3 4 5 6 7 Very natural

7. Humanlike Voice: To what extent did this voice sound like a human?

   Nothing like a human 1 2 3 4 5 6 7 Just like a human

8. Voice Quality: Did the voice sound harsh, raspy, or strained?

   Significantly harsh/raspy 1 2 3 4 5 6 7 Normal quality

9. Rhythm: Did the rhythm of the speech sound natural?

   Unnatural or mechanical 1 2 3 4 5 6 7 Natural rhythm

10. Intonation: Did the intonation pattern of sentences sound smooth and natural?

    Abrupt or abnormal 1 2 3 4 5 6 7 Smooth or natural

E.2 List of proposed features

Prosody Features	Spectral Features	Cepstral Features	Voicing Features
audspec_lengthL1norm_sma	pcm_fftMag_psySharpness_sma	pcm_fftMag_mfcc_sma[1]	jitterDDP_sma
F0final_sma	pcm_fftMag_fband250-650_sma	.	jitterLocal_sma
pcm_zcr_sma	pcm_fftMag_fband1000-4000_sma	.	shimmerLocal_sma
	pcm_fftMag_spectralRollOff25.0_sma	pcm_fftMag_mfcc_sma[16]	logHNR_sma
	pcm_fftMag_spectralRollOff50.0_sma		voicingFinalUnclipped_sma
	pcm_fftMag_spectralRollOff75.0_sma		
	pcm_fftMag_spectralRollOff90.0_sma		
	pcm_fftMag_spectralEntropy_sma		
	pcm_fftMag_spectralFlatness_sma		
	pcm_fftMag_spectralFlux_sma		
	pcm_fftMag_spectralHarmonicity_sma		
	pcm_fftMag_spectralKurtosis_sma		
	pcm_fftMag_spectralSkewness_sma		
	pcm_fftMag_spectralVariance_sma		

E.3 Mean and Standard Deviation of proposed features

Table E.1 List of energy, spectral and voicing related LLD with higher significant differences (Mann-Whitney test with p-value < 0.001) extracted from CMU Models.

Features	**CMU-ORIG**	**CMU-HMM**	**CMU-CLU**	**CMU-DNN**	**CMU-USS**
Loudness	0.928(0.357)	0.493(0.271)	0.941(0.323)	0.901(0.301)	0.949(0.367)
Pitch	189.846(58.557)	177.862(54.412)	190.084(50.226)	195.95(49.16)	191.45(59.846)
ZCR	0.12(0.068)	0.116(0.071)	0.138(0.072)	0.153(0.061)	0.121(0.066)
Psychoacoustic	0.735(0.315)	0.675(0.366)	0.751(0.354)	0.661(0.309)	0.74(0.317)
SBE (250-650Hz)	0.167(0.212)	0.034(0.075)	0.18(0.196)	0.106(0.123)	0.174(0.219)
SBE (1-4kHz)	0.531(0.534)	0.774(1.395)	1.12(1.411)	0.513(0.433)	0.547(0.533)
Spectral RoP (25%)	549.92(390.44)	546.939(459.566)	617.074(465.084)	489.775(367.398)	553.259(395.93)
Spectral RoP (50%)	764.013(531.252)	745.427(593.635)	826.697(590.666)	689.095(507.645)	767.385(535.296)
Spectral RoP (75%)	1085.537(673.674)	1024.764(717.245)	1140.971(689.706)	949.431(638.214)	1093.503(679.087)
Spectral RoP (90%)	1636.698(832.379)	1430.282(850.404)	1687.207(817.161)	1427.826(794.101)	1651.614(835.86)
Spectral Entropy	3.568(0.608)	3.159(0.656)	3.354(0.666)	3.694(0.558)	3.588(0.601)
Spectral Flatness	0.026(0.028)	0.021(0.018)	0.02(0.016)	0.027(0.029)	0.026(0.026)
Spectral Flux	0.301(0.141)	0.23(0.179)	0.327(0.119)	0.254(0.098)	0.308(0.145)
Spectral Harmonicity	0.538(0.428)	0.57(0.851)	0.787(0.702)	0.483(0.294)	0.555(0.437)
Spectral Kurtosis	112.341(126.046)	566.286(673.203)	399.782(506.024)	77.504(52.36)	103.187(117.85)
Spectral Skewness	5.659(2.65)	10.649(6.05)	8.961(5.423)	5.181(1.7)	5.437(2.581)
Spectral Variance	826464.878(590285.4)	704440.492(542941.897)	851551.758(456434.595)	714205.123(547582.003)	827666.929(585041.665)
MFCC-1	17.984(8.107)	19.397(8.795)	15.42(7.795)	21.067(7.558)	17.913(8.206)
MFCC-2	0.065(11.858)	7.013(10.86)	7.14(12.056)	2.313(10.745)	-0.432(11.988)
MFCC-3	12.021(10.754)	20.412(10.581)	23.296(12.214)	15.266(10.518)	12.076(10.926)
MFCC-4	-29.6(11.69784)	-22.061(11.268)	-31.4(10.412)	-27.813(9.85)	-30.14(11.787)
MFCC-5	-5.357(11.702)	-5.254(14.128)	-2.237(13.524)	-3.138(10.478)	-5.326(11.935)
MFCC-6	-20.551(10.415)	-17.477(10.95)	-15.807(10.614)	-16.591(7.97)	-20.362(10.666)
MFCC-7	2.262(9.333)	1.95(11.248)	3.764(11.089)	6.62(7.842)	2.403(9.366)
MFCC-8	-17.778(12.184)	-14.772(14.132)	-28.624(12.385)	-15.043(9.803)	-17.984(12.266)
MFCC-9	-12.121(10.336)	-11.277(9.965)	-17.162(10.314)	-8.268(7.246)	-11.995(10.265)
MFCC-10	-13.638(8.617)	-9.795(9.878)	-9.329(8.849)	-10.08(7.061)	-13.539(8.655)
MFCC-11	-17.679(8.497)	-17.52(9.103)	-25.008(9.069)	-15.704(5.811)	-17.836(8.439)
MFCC-12	-7.536(6.94)	-6.194(7.353)	-8.836(6.534)	-4.823(4.682)	-7.477(6.897)
MFCC-13	-5.868(6.671)	-7.378(7.056)	-12.476(6.279)	-5.546(4.769)	-6.075(6.685)
MFCC-14	-9.737(7.652)	-10.244(7.088)	-11.126(6.687)	-11.054(5.919)	-9.789(7.703)
MFCC-15	-1.568(6.676)	-2.846(5.754)	-2.959(5.676)	-2.471(5.291)	-1.559(6.69)
MFCC-16	-3.062(6.853)	-4.2(5.938)	-5.982(6.032)	-3.686(6.677)	-3.1(6.941)
Jitter Local	0.027(0.018)	0.032(0.021)	0.028(0.019)	0.025(0.017)	0.027(0.018)
Jitter δ	0.021(0.017)	0.027(0.02)	0.022(0.015)	0.02(0.014)	0.021(0.016)
Shimmer	0.119(0.069)	0.151(0.063)	0.117(0.055)	0.108(0.057)	0.117(0.069)
logHNR	-11.898(26.177)	-17.219(25.124)	-11.141(23.251)	-6.326(24.317)	-11.241(25.883)
Voicing Probability	0.747(0.093)	0.665(0.135)	0.735(0.089)	0.757(0.085)	0.748(0.092)
Delta Regression Coefficient					
Loudness	0.034(0.049)	0.017(0.029)	0.037(0.046)	0.035(0.049)	0.033(0.051)
Pitch	-0.013(0.177)	0.035(0.25)	-0.043(0.17)	-0.006(0.162)	-0.01(0.24)
ZCR	0.001(0.007)	0.001(0.006)	0.001(0.006)	0.002(0.007)	0.001(0.007)
Psychoacoustic	0.009(0.029)	0.008(0.033)	0.008(0.029)	0.008(0.029)	0.007(0.032)
SBE (250-650Hz)	0.026(0.065)	0.072(0.218)	0.087(0.198)	0.031(0.062)	0.026(0.071)
SBE (1-4kHz)	0.011(0.025)	0.001(0.01)	0.014(0.022)	0.008(0.015)	0.01(0.025)
Spectral RoP (25%)	5.794(35.02)	6.033(42.642)	5.079(33.626)	5.6(31.318)	3.251(39.344)
Spectral RoP (50%)	7.46(46.951)	9.008(54.519)	5.231(47.568)	6.539(45.965)	4.399(53.784)
Spectral RoP (75%)	15.448(70.025)	12.917(67.402)	10.522(65.88)	9.95(63.904)	11.111(74.488)
Spectral RoP (90%)	29.484(98.193)	18.504(82.0)	25.717(88.817)	24.234(93.212)	25.029(99.05)
Spectral Entropy	0.019(0.077)	0.015(0.067)	0.026(0.072)	0.017(0.078)	0.013(0.083)
Spectral Flatness	-0.0(0.003)	-0.0(0.002)	0.0(0.002)	-0.0(0.003)	-0.001(0.003)
Spectral Flux	0.014(0.017)	0.011(0.018)	0.015(0.017)	0.012(0.016)	0.014(0.018)
Spectral Harmonicity	0.027(0.053)	0.035(0.117)	0.056(0.105)	0.026(0.045)	0.027(0.057)
Spectral Kurtosis	-11.29(28.329)	-44.667(120.56)	-50.201(108.855)	-7.172(12.026)	-8.937(27.565)
Spectral Skewness	-0.276(0.529)	-0.449(0.978)	-0.687(1.079)	-0.216(0.343)	-0.217(0.494)
Spectral Variance	5041.472(68175.251)	1366.077(53727.963)	6485.618(57914.734)	4294.615(59976.578)	2485.657(68440.089)
MFCC-1	-0.162(0.857)	-0.096(0.78)	-0.157(0.757)	-0.126(0.787)	-0.125(0.87)
MFCC-2	-0.814(1.324)	-0.759(1.075)	-0.923(1.287)	-0.762(1.207)	-0.759(1.283)
MFCC-3	0.173(1.154)	0.268(1.048)	0.378(1.323)	0.185(1.129)	0.224(1.161)
MFCC-4	-1.187(1.725)	-1.34(1.655)	-1.241(1.905)	-1.182(1.613)	-1.18(1.767)
MFCC-5	-0.372(1.107)	-0.745(1.303)	-0.306(1.326)	-0.278(1.018)	-0.433(1.167)
MFCC-6	-0.344(1.189)	-0.594(1.164)	-0.304(1.109)	-0.32(1.039)	-0.398(1.273)
MFCC-7	0.151(1.084)	0.056(1.245)	0.11(1.172)	0.315(0.99)	0.106(1.084)
MFCC-8	-0.305(1.222)	-0.486(1.359)	-0.342(1.178)	-0.192(0.946)	-0.336(1.284)
MFCC-9	0.011(0.946)	-0.162(0.957)	0.028(0.869)	0.04(0.742)	-0.055(1.006)
MFCC-10	-0.027(0.967)	0.134(1.122)	0.111(0.905)	0.042(0.787)	-0.092(0.993)
MFCC-11	-0.365(1.009)	-0.549(1.156)	-0.563(1.168)	-0.304(0.787)	-0.427(1.126)
MFCC-12	0.113(0.799)	0.113(0.836)	0.064(0.683)	0.124(0.567)	0.115(0.84)
MFCC-13	-0.177(0.789)	-0.198(0.806)	-0.306(0.793)	-0.106(0.552)	-0.222(0.818)
MFCC-14	-0.028(0.794)	0.003(0.768)	-0.112(0.682)	0.068(0.603)	-0.026(0.829)
MFCC-15	0.189(0.715)	0.246(0.65)	0.024(0.612)	0.189(0.61)	0.192(0.717)
MFCC-16	0.148(0.67)	0.123(0.519)	0.041(0.581)	0.201(0.597)	0.139(0.666)
Jitter Local	-0.0(0.0)	-0.0(0.0)	-0.0(0.0)	-0.0(0.0)	-0.0(0.0)
Jitter δ	-0.0(0.0)	-0.0(0.0)	-0.0(0.0)	-0.0(0.0)	-0.0(0.0)
Shimmer	-0.001(0.002)	-0.001(0.001)	-0.001(0.001)	-0.001(0.002)	-0.001(0.002)
logHNR	0.344(0.579)	0.431(0.685)	0.364(0.618)	0.364(0.612)	0.369(0.637)
Voicing Probability	0.001(0.002)	0.001(0.001)	0.001(0.002)	0.001(0.002)	0.001(0.002)

† Values in the table $p(q)$: p is mean and q is standard deviation.

Table E.2 List of energy, spectral and voicing related LLD with higher significant differences (Mann-Whitney test with p-value < 0.001) extracted from IITM Models.

Features	**IITM-ORIG**	**IITM-HMM**	**IITM-CLU**	**IITM-DNN**	**IITM-USS**
Loudness	0.933(0.333)	0.214(0.22)	0.791(0.267)	0.702(0.229)	0.935(0.344)
Pitch	144.929(42.978)	129.166(50.364)	147.55(35.355)	152.438(35.798)	143.915(44.946)
ZCR	0.091(0.069)	0.138(0.128)	0.105(0.083)	0.106(0.063)	0.097(0.085)
Psychoacoustic	0.575(0.345)	0.639(0.507)	0.571(0.417)	0.512(0.319)	0.601(0.39)
SBE (250-650Hz)	0.137(0.229)	0.007(0.033)	0.078(0.124)	0.038(0.055)	0.135(0.225)
SBE (1-4kHz)	1.235(1.172)	0.231(0.599)	0.699(0.603)	0.612(0.598)	1.248(1.219)
Spectral RoP (25%)	416.465(403.216)	556.676(653.426)	523.102(575.106)	349.795(340.146)	446.972(460.493)
Spectral RoP (50%)	647.327(573.393)	741.511(790.355)	667.073(671.919)	564.128(525.653)	679.266(631.079)
Spectral RoP (75%)	887.55(704.026)	990.923(916.845)	875.938(757.728)	808.429(664.781)	928.321(774.689)
Spectral RoP (90%)	1148.045(813.453)	1290.936(1030.263)	1137.642(846.096)	1060.082(785.361)	1199.592(899.657)
Spectral Entropy	3.34(0.607)	3.941(3.235)	2.924(0.661)	3.468(0.557)	3.406(1.061)
Spectral Flatness	0.019(0.023)	0.029(0.036)	0.015(0.015)	0.02(0.025)	0.025(0.034)
Spectral Flux	0.384(0.133)	0.121(0.18)	0.374(0.092)	0.246(0.08)	0.392(0.14)
Spectral Harmonicity	1.125(0.792)	0.283(0.62)	1.291(0.733)	0.632(0.427)	1.124(0.82)
Spectral Kurtosis	249.573(299.59)	752.832(896.929)	920.961(843.221)	133.844(87.203)	249.273(297.609)
Spectral Skewness	7.534(3.592)	11.81(7.577)	14.146(6.799)	6.105(2.227)	7.521(3.654)
Spectral Variance	545891.893(633018.486)	583248.296(572941.533)	500377.55(460000.119)	522455.593(642910.764)	585230.444(687909.473)
MFCC-1	-3.676(7.893)	-4.202(7.718)	-1.289(7.548)	-2.181(6.703)	-2.806(7.822)
MFCC-2	-11.097(7.698)	-11.854(7.382)	-14.515(7.109)	-10.453(6.43)	-12.139(7.925)
MFCC-3	-1.757(6.1)	-2.064(6.263)	-2.561(5.806)	-0.759(4.51)	-0.793(6.259)
MFCC-4	-6.387(6.493)	-7.025(5.494)	-9.837(4.279)	-6.822(4.395)	-7.282(6.656)
MFCC-5	-8.329(4.691)	-8.771(4.698)	-9.783(3.688)	-7.922(3.322)	-7.62(4.763)
MFCC-6	-2.595(4.911)	-3.774(4.61)	-2.682(3.758)	-2.359(3.716)	-3.289(5.037)
MFCC-7	-5.71(3.868)	-5.352(3.681)	-8.706(2.911)	-5.099(2.689)	-5.23(3.949)
MFCC-8	30.664(9.408)	27.029(11.666)	28.457(9.285)	32.607(9.313)	30.277(9.8)
MFCC-9	9.066(10.791)	14.853(9.749)	18.539(9.869)	10.361(10.13)	9.347(10.98)
MFCC-10	21.464(14.216)	22.136(13.05)	31.242(14.303)	23.428(13.079)	20.701(14.049)
MFCC-11	-10.279(9.379)	-2.461(8.893)	-4.812(9.465)	-10.543(8.732)	-9.704(9.578)
MFCC-12	-3.501(12.277)	-3.188(11.76)	-0.467(12.444)	-4.281(11.62)	-4.611(12.28)
MFCC-13	-8.022(10.179)	-2.722(9.084)	2.398(8.8)	-8.178(8.576)	-7.187(10.066)
MFCC-14	-11.167(10.263)	-10.118(10.197)	-10.329(9.315)	-10.401(9.167)	-12.138(10.627)
MFCC-15	-9.193(10.985)	-6.274(11.45)	-15.187(10.131)	-7.568(9.346)	-8.257(10.907)
MFCC-16	-2.681(7.758)	-2.273(7.313)	-2.835(6.572)	0.14(5.53)	-3.816(7.778)
Jitter Local	0.027(0.019)	0.025(0.019)	0.026(0.018)	0.027(0.018)	0.026(0.02)
Jitter δ	0.019(0.015)	0.019(0.015)	0.02(0.014)	0.02(0.014)	0.019(0.015)
Shimmer	0.125(0.077)	0.136(0.075)	0.129(0.073)	0.129(0.073)	0.123(0.082)
logHNR	-12.758(24.937)	-22.267(30.073)	-9.299(23.156)	-5.855(23.354)	-13.502(25.825)
Voicing Probability	0.74(0.063)	0.575(0.199)	0.727(0.065)	0.742(0.061)	0.735(0.077)
Delta Regression Coefficient					
Loudness	0.016(0.041)	0.003(0.014)	0.017(0.036)	0.018(0.032)	0.014(0.043)
Pitch	-0.011(0.158)	-0.016(0.142)	-0.006(0.102)	0.004(0.132)	-0.012(0.175)
ZCR	-0.001(0.007)	-0.003(0.015)	-0.001(0.008)	-0.001(0.007)	-0.001(0.009^{th})
Psychoacoustic	-0.006(0.043)	-0.004(0.052)	-0.005(0.046)	-0.003(0.035)	-0.007(0.052)
SBE (250-650Hz)	0.045(0.125)	0.003(0.077)	0.025(0.058)	0.025(0.06)	0.04(0.128)
SBE (1-4kHz)	0.002(0.015)	-0.0(0.002)	0.002(0.008)	0.001(0.004)	0.001(0.015)
Spectral RoP (25%)	-3.477(48.017)	-0.97(66.794)	-4.59(65.679)	1.047(34.684)	-4.09(57.324)
Spectral RoP (50%)	-9.793(75.282)	-5.529(81.444)	-9.218(78.358)	-3.806(58.004)	-10.597(87.561)
Spectral RoP (75%)	-15.591(94.749)	-9.714(97.837)	-11.8(89.281)	-9.609(78.935)	-17.143(113.164)
Spectral RoP (90%)	-23.798(117.348)	-17.316(118.285)	-17.79(109.665)	-17.261(100.19)	-26.042(139.286)
Spectral Entropy	-0.022(0.105)	-0.071(0.56)	-0.02(0.094)	-0.018(0.089)	-0.017(0.409)
Spectral Flatness	-0.001(0.004)	-0.001(0.005)	-0.001(0.003)	-0.001(0.004)	-0.001(0.007)
Spectral Flux	0.011(0.02)	0.003(0.012)	0.012(0.016)	0.009(0.013)	0.013(0.022)
Spectral Harmonicity	0.034(0.085)	0.005(0.061)	0.033(0.077)	0.022(0.048)	0.03(0.087)
Spectral Kurtosis	-11.79(50.375)	-40.324(151.476)	-60.341(152.776)	-4.375(14.86)	-11.972(54.28)
Spectral Skewness	-0.074(0.59)	-0.311(1.187)	-0.383(1.104)	-0.027(0.346)	-0.059(0.629)
Spectral Variance	-19239.046(83399.575)	-22105.093(77133.873)	-13669.214(61738.218)	-15245.709(75744.255)	-22135.78(101462.362)
MFCC-1	0.308(1.114)	0.374(1.207)	0.305(1.094)	0.298(1.023)	0.3(1.196)
MFCC-2	-0.115(1.125)	-0.109(1.007)	-0.145(1.008)	-0.092(1.02)	-0.113(1.187)
MFCC-3	0.448(1.314)	0.494(1.284)	0.51(1.341)	0.501(1.23)	0.458(1.406)
MFCC-4	-0.565(1.425)	-0.497(1.411)	-0.549(1.437)	-0.758(1.28)	-0.505(1.454)
MFCC-5	-0.498(1.157)	-0.588(1.172)	-0.473(1.148)	-0.405(1.045)	-0.495(1.199)
MFCC-6	-0.036(1.042)	-0.096(0.967)	0.069(0.915)	-0.086(0.919)	-0.063(1.153)
MFCC-7	-0.351(1.179)	-0.421(1.14)	-0.387(1.058)	-0.363(1.01)	-0.351(1.245)
MFCC-8	-0.22(1.05)	-0.214(1.101)	-0.208(0.938)	-0.182(0.874)	-0.187(1.101)
MFCC-9	-0.033(0.909)	0.12(0.859)	0.17(0.744)	-0.034(0.645)	-0.015(0.939)
MFCC-10	-0.018(0.898)	-0.067(0.876)	0.032(0.84)	0.073(0.735)	-0.03(0.935)
MFCC-11	-0.371(0.988)	-0.325(0.894)	-0.345(0.932)	-0.318(0.781)	-0.367(1.062)
MFCC-12	0.062(0.741)	0.094(0.712)	0.136(0.62)	0.082(0.532)	0.07(0.807)
MFCC-13	-0.071(0.736)	-0.043(0.637)	-0.031(0.489)	-0.106(0.535)	-0.057(0.762)
MFCC-14	-0.219(0.681)	-0.311(0.691)	-0.199(0.572)	-0.167(0.488)	-0.216(0.687)
MFCC-15	-0.065(0.58)	-0.086(0.595)	-0.017(0.434)	-0.089(0.431)	-0.064(0.622)
MFCC-16	-0.069(0.482)	-0.124(0.481)	-0.091(0.345)	-0.099(0.348)	-0.074(0.511)
Jitter Local	-0.0(0.0)	-0.0(0.0)	-0.0(0.0)	-0.0(0.0)	-0.0(0.0)
Jitter δ	-0.0(0.0)	-0.0(0.0)	-0.0(0.0)	-0.0(0.0)	-0.0(0.0)
Shimmer	-0.001(0.002)	-0.001(0.002)	-0.001(0.002)	-0.001(0.002)	-0.001(0.002)
logHNR	0.336(0.613)	0.515(0.858)	0.363(0.633)	0.342(0.612)	0.352(0.656)
Voicing Probability	0.001(0.002)	0.0(0.001)	0.001(0.002)	0.001(0.002)	0.001(0.002)

† Values in the table $p(q)$: p is mean and q is standard deviation.