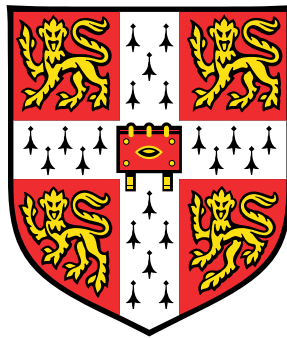


Discriminative Methods for Statistical Spoken Dialogue Systems



Matthew S. Henderson

Department of Engineering
University of Cambridge

This dissertation is submitted for the degree of
Doctor of Philosophy

Churchill College

February 2015

DECLARATION

I hereby declare that except where specific reference is made to the work of others, the contents of this dissertation are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other university. This dissertation is my own work and contains nothing which is the outcome of work done in collaboration with others, except as specified in the text and Acknowledgements. This dissertation contains roughly 31,000 words including appendices, bibliography, footnotes, tables and equations and has 47 figures and tables. Some of the work presented here was published in the Special Interest Group on Discourse and Dialogue (Henderson et al., 2014b, 2013, 2014a) and the IEEE workshop on Spoken Language Technology (Henderson et al., 2014c, 2012, 2014d).

Matthew S. Henderson
February 2015

ACKNOWLEDGEMENTS

I am very grateful to Steve Young for his supervision throughout my PhD. Steve has devoted a lot of time in providing me with excellent guidance and considered feedback. He has taught me a lot about clear thinking and effective research, and he has in general been an inspiration. I am also thankful to Blaise Thomson, my advisor, for his direction and support, and for helping to make this an enjoyable process.

It has been a privilege to work with my lab-mates in the Cambridge Dialogue Systems group, whose help in developing the work presented here has been invaluable. I would like to thank all of those who have been a part of the group beside me, in particular Milica Gašić, Pirros Tsiakoulis, Dongho Kim, Jorge Prombonas, Kai Yu, Catherine Breslin, Martin Szummer, David Vandyke, Nikola Mrksic, Tsung-Hsien Wen, and Pei-Hao Su. I would also like to sincerely thank Anna Langley and Patrick Gosling for their computing support, as well as Janet Milne and Rachel Fogg for their administrative support.

Outside of the department, I would like to acknowledge Jason Williams for his invaluable guidance. His encouragement and generous support have resulted in gratifying collaborations that have helped to define this thesis.

I acknowledge the EPSRC for funding, as well as the Google Doctoral Fellowship, which has provided me with excellent support, and an outstanding mentor in Brian Strope.

Lastly I would like to offer a heartfelt thankyou to my parents, to Nicky, and to Gillian for their help in proof-reading, and moreover for their encouragement and understanding support.

ABSTRACT

Dialogue promises a natural and effective method for users to interact with and obtain information from computer systems. Statistical spoken dialogue systems are able to disambiguate in the presence of errors by maintaining probability distributions over what they believe to be the state of a dialogue. However, traditionally these distributions have been derived using generative models, which do not directly optimise for the criterion of interest and cannot easily exploit arbitrary information that may potentially be useful. This thesis presents how discriminative methods can overcome these problems in Spoken Language Understanding (SLU) and Dialogue State Tracking (DST).

A robust method for SLU is proposed, based on features extracted from the full posterior distribution of recognition hypotheses encoded in the form of word confusion networks. This method uses discriminative classifiers, trained on unaligned input/output pairs. Performance is evaluated on both an off-line corpus, and on-line in a live user trial. It is shown that a statistical discriminative approach to SLU operating on the full posterior ASR output distribution can substantially improve performance in terms of both accuracy and overall dialogue reward. Furthermore, additional gains can be obtained by incorporating features from the system's output.

For DST, a new word-based tracking method is presented that maps directly from the speech recognition results to the dialogue state without using an explicit semantic decoder. The method is based on a recurrent neural network structure that is capable of generalising to unseen dialogue state hypotheses, and requires very little feature engineering. The method is evaluated in the second and third Dialog State Tracking Challenges, as well as in a live user trial. The results demonstrate consistently high performance across all of the off-line metrics and a substantial increase in the quality of the dialogues in the live trial. The proposed method is shown to be readily applied to expanding dialogue domains, by exploiting robust features and a new method for online unsupervised adaptation. It is shown how the neural network structure can be adapted to output structured joint distributions, giving an improvement over estimating the dialogue state as a product of marginal distributions.

TABLE OF CONTENTS

List of figures	xiii
List of tables	xv
1 Introduction	1
1.1 Thesis Outline and Contributions	3
2 Overview of Spoken Dialogue Systems	5
2.1 Automatic Speech Recognition	6
2.2 Spoken Language Understanding	8
2.3 Dialogue State Tracking	11
2.4 Dialogue Management	13
2.4.1 Statistical Dialogue Systems	13
2.5 Natural Language Generation and Speech Synthesis	14
3 Spoken Language Understanding	17
3.1 Methods for Spoken Language Understanding	18
3.1.1 Using a Robust Grammar	18
3.1.2 Using a Conditional Random Field	18
3.1.3 Using Discriminative Classifiers	22
3.2 In-Car Spoken Language Understanding Corpus	23
3.2.1 Labelling	24
3.2.2 Metrics	25
3.3 Evaluation of Spoken Language Understanding Methods	26
3.4 Conclusions	28
4 Understanding Speech Recogniser Output	29
4.1 Semantic Decoder Configurations	30

4.1.1	Weighted Sum Features	32
4.1.2	Word Confusion Network Features	32
4.1.3	Dialogue Context Features	33
4.2	Off-line Evaluation	33
4.3	Live User Evaluation	36
4.3.1	Experimental Setup	36
4.3.2	Results	38
4.4	Conclusions	40
5	Dialogue State Tracking	41
5.1	Formulation of Dialogue State Tracking	42
5.2	Baseline Methods for Tracking	44
5.2.1	SLU Evidence Observations	44
5.2.2	One-best Baseline	45
5.2.3	Focus Baseline	46
5.2.4	HWU Baseline	47
5.3	Generative Tracking with Dynamic Bayesian Networks	47
5.4	Discriminative Dialogue State Tracking	50
5.4.1	Using Static Classifiers	51
5.4.2	Using Conditional Random Fields	53
5.5	Discriminative Tracking with Recurrent Neural Networks	53
5.5.1	Feature Representation	53
5.5.2	Generalisation to Unseen Dialogue States	55
5.5.3	Recurrent Neural Network Structure	55
5.5.4	Requested Slots and Search Method	59
5.5.5	Key Parameters	60
5.6	Direct Word-based Tracking	60
5.7	Training Recurrent Neural Network Trackers	61
5.7.1	Denoising Autoencoder Initialisation	63
5.7.2	Shared Initialisation	64
5.7.3	Model Combination	64
5.7.4	Training Set Size	64
5.8	Conclusions	64
6	Evaluation of Dialogue State Tracking	67
6.1	Dialog State Tracking Challenges	68
6.1.1	Data	69

6.1.2	Evaluation Metrics	72
6.1.3	Oracle Tracker	73
6.2	The Second Dialog State Tracking Challenge Evaluation	74
6.3	The Third Dialog State Tracking Challenge Evaluation	76
6.4	Changing Goal Constraints	79
6.5	Live User Evaluation	81
6.5.1	Experimental Setup	83
6.5.2	Results	86
6.6	Conclusions	86
7	Unsupervised Adaptation for RNN Dialogue State Tracking	91
7.1	Online Adaptation	92
7.1.1	Criterion for Unsupervised Adaptation	92
7.1.2	Schedule for Online Adaptation	94
7.2	Evaluation	94
7.2.1	Adaptation with Missing Labels	94
7.2.2	Adaptation to New Domains	96
7.3	Conclusions	97
8	Structured Outputs for RNN Dialogue State Tracking	99
8.1	Related work	100
8.2	Structured Joint Predictions	100
8.2.1	Conditional Structured Joint Predictions	102
8.3	Evaluation	104
8.4	Conclusions	106
9	Conclusions	109
9.1	Current Limitations and Future Work	111
	Appendix A Slot-based Dialogue Domains	115
	Appendix B Dialogue Act Format	117
	Appendix C Matching Strings to the Domain with Aliases	123
C.1	Aliases	123
C.1.1	Aliases for DSTC 3 evaluation	123
C.1.2	Supplementary Aliases for CRF Tagging	127

Appendix D Learning Neural Networks	129
D.1 Multi-layer Perceptrons	129
D.2 Parameter Initialisation	130
D.2.1 Random Initialisation	130
D.2.2 Denoising Autoencoder Initialisation	131
D.3 Recurrent Neural Networks	131
D.4 Stochastic Gradient Descent	132
D.5 Regularisation	134
 Glossary	 135
 Acronyms	 139
 References	 141

LIST OF FIGURES

2.1	The pipeline architecture of a typical dialogue system	5
2.2	Representations of the speech recognition posterior distribution	7
2.3	Example of aligned and unaligned labels in ATIS	10
3.1	Example BIO encodings in restaurant information Domain	19
4.1	Semantic decoder configurations.	31
4.2	Illustrative example of semantic decoder output.	36
4.3	Regressions of F-score and ICE against WER.	37
4.4	Regressions of success, reward and number of turns against WER.	39
5.1	Example dialogue with dialogue state labels	43
5.2	Calculation of SLU observations	45
5.3	Dynamic Bayesian Network structure for Dialogue State Tracking	49
5.4	Calculation of new output and memory in RNN structure	56
5.5	Recurrent Neural Network structure for Dialogue State Tracking	57
5.6	Delexicalised Recurrent Neural Network structure for Dialogue State Tracking	58
5.7	Example of feature extraction for RNN dialogue state trackers	62
5.8	The effect of combining multiple RNN trackers	65
5.9	Effect of the size of the training set for RNN trackers	65
6.1	Example dialogue illustrating a change in goal constraint	82
6.2	Gaussian process reinforcement learning curves.	89
7.1	Example dialogue where user guidance can be exploited in unsupervised learning	93
7.2	Online unsupervised adaptation with missing labels	95

LIST OF TABLES

1.1	Example scenario where statistical dialogue theory could help	2
3.1	Statistics of the in-car Spoken Language Understanding Corpus	24
3.2	Training and testing on transcriptions	27
3.3	Training on transcriptions and noisy ASR output	27
4.1	Off-line evaluation of semantic decoders.	34
4.2	Results of user trial.	40
5.1	Extracting n -gram features from dialogue acts	54
5.2	The effect of initialisation techniques for RNN training	63
6.1	Goal changes in DSTC 2 and 3 data	70
6.2	Simple statistics of the DSTC 2 and 3 datasets	71
6.3	WER and F-score statistics for DSTC 2 and 3 data	71
6.4	DSTC 2 results	75
6.5	DSTC 3 results	78
6.6	Breakdown of performance on dialogues with and without goal changes . .	79
6.7	Breakdown of performance on dialogues with and without goal changes in the DSTC 2 test set	80
6.8	Results of live user trial using learnt policies.	87
6.9	Results across ASR conditions with statistical significance tests	88
7.1	Online unsupervised adaptation to new domains	96
7.2	Performance of adapted model on each slot	97
8.1	Effect of using structured joint output	104
8.2	None accuracy for SLU-based RNNs	105
8.3	None accuracy for word-based RNNs	105

A.1	Slots in restaurant information and tourist information domains.	116
B.1	Dialogue act types for user actions.	119
B.2	Dialogue act types for system actions.	120
B.3	Examples of dialogue acts.	121

CHAPTER 1

INTRODUCTION

Research on spoken dialogue systems seeks to create computer systems that can hold a conversation using natural language, just like a real human conversational partner. This is an attractive proposition, particularly with the increasing pervasiveness of smart phones and ubiquitous smart systems such as wearable devices, in-car computers, and smart homes. These systems do not necessarily have large screens or conventional keyboard and mouse input methods, but often have microphones and speakers (Cohen and Oviatt, 1994).

In his *Discourse on the Method*, Descartes predicted that such a computer system could never be possible, asserting that “it is not conceivable that such a machine should produce different arrangements of words so as to give an appropriately meaningful answer to whatever is said in its presence, as the dumbest of men can do.” Turing’s imitation game seeks to test this hypothesis by asking human participants to judge whether they are speaking to another human or a computer system (Turing et al., 1952).

The Loebner Prize (Epstein, 1992), an instantiation of Turing’s imitation game for text-based dialogue systems, has not yet had a successful claimant. Futurist Ray Kurzweil predicts there will be no winner until the 2020’s (Kurzweil, 2006). The current state of the art is so far from this point that work on this topic has faced serious criticism (Sundman, 2003). This thesis focusses on spoken dialogue systems in much more constrained domains, whose purpose is to help the user with a specific and well defined task.

Such constrained task-oriented systems have been deployed for a variety of tasks. Applications include travel itinerary information (Norton et al., 1990), in-car information and navigation (Becker et al., 2006), robot tour guides (Faber et al., 2009), and even robot bar tenders (Foster et al., 2012). This thesis uses tourist information as an example application, where users of the system can search for and ask about venues like restaurants in a city according to their constraints (e.g. by cuisine or part of town).

When exposed to the public, a spoken dialogue system may encounter a variety of dif-

difficulties. The system may see large variation in the users of the system, such as differing levels of familiarity with dialogue systems and varying ways of speaking. Systems may also be subject to high noise compared to lab environments, especially those deployed on mobile devices. The task of a dialogue system to understand what the user wants at any point in a conversation is consequently difficult, but still essential for a successful dialogue.

A statistical framework is used in this thesis, where the system maintains a probability distribution over what the user wants (the *state* of the dialogue). This thesis argues that *discriminative* modelling is an effective technique for estimating these distributions, and shows dialogue systems can benefit from these improved estimations.

Utterance		Spoken Language Understanding	
System	How may I help you?		
		<i>train-station</i>	0.2
User	Where are the petrol stations in Cambridge?	<i>petrol-station</i>	0.1
		<i>Other</i>	0.7
System	Could you please repeat that?		
		<i>train-station</i>	0.2
User	Where are the petrol stations in Cambridge?	<i>petrol-station</i>	0.1
		<i>Other</i>	0.7
System	Could you please repeat that?		
	...		

Table 1.1: An example scenario with a fictional in-car dialogue system where statistical dialogue theory could help. The rightmost columns present the output of a Spoken Language Understanding component, which takes the noisy speech of the user and attempts to discern what type of destination the user is seeking. Here at each turn the key word ‘*petrol*’ is not recognised. A conventional dialogue system might only consider the top scoring hypothesis, *train-station*, but reject it as it has a low confidence score (0.2). The scenario is detailed further in the text.

Table 1.1 presents a scenario where using a distribution rather than a single top hypothesis could help in a dialogue. Faced with identical inputs, the computer system in this scenario gives identical outputs, bringing the interaction into what is termed a downward spiral of errors (Bohus, 2007). A statistical dialogue system maintains uncertainty about everything the user has said, and exploits the sequential nature of dialogue to disambiguate in the presence of errors. In the example scenario, the noisy input might lead to some weight in the system’s distribution over internal states being assigned to two competing hypotheses – *petrol station* and *train station*. Perhaps the system asks, “*You are looking for the train station?*” and the driver replies “*No.*” The only remaining valid hypothesis, *petrol station*,

would then be correctly identified.

Previously proposed methods for estimating the dialogue’s state generally have applied highly application-specific models to dialogue systems, for example *generative* probabilistic models to jointly model the hidden dialogue state and the observations (Horvitz and Paek, 1999; Williams and Young, 2005; Young, 2000). This kind of approach has advantages, but crucially these models cannot easily be used to exploit arbitrary information or *features*. The key in finding significant improvements is the use of discriminative modelling. Discriminative models directly estimate the probability distribution of the important variables in a problem, such as dialogue acts or dialogue states, given a potentially large set of arbitrary features. Well understood and general models such as Support Vector Machines (SVMs) and artificial neural networks can then be exploited.

1.1 Thesis Outline and Contributions

Once a user has spoken to a dialogue system, the audio signal is then passed through a speech recogniser to give a distribution over words. A Spoken Dialogue System (SDS) attempts to understand the words and how they affect the current state of the dialogue. Typically, this task is split into two steps – Spoken Language Understanding (SLU) converts the words from the speech recognition into a semantic representation, and Dialogue State Tracking (DST) takes the output of the SLU and estimates the new state of the dialogue.

After an introductory chapter (chapter 2), which lays the foundations of statistical spoken dialogue systems and sets the context of the work, this thesis is split into two main sections. Chapters 3 and 4 focus on SLU, and chapters 5, 6, 7, and 8 look at DST. Below is a description of each of these chapters and their contributions.

Chapter 3, Spoken Language Understanding. This chapter presents and compares three methods for SLU, contrasting a grammar-based method to two discriminative statistical approaches. The two discriminative approaches demonstrate a contrast between treating SLU as a sequential labelling problem over sequences of words, and alternatively as a flat labelling problem. A corpus of noisy in-car dialogues is presented, which is labelled for the purpose of evaluating the three methods. After the analysis of an off-line evaluation, the Semantic Tuple Classifier (STC) method is selected as a promising method for extension in the following chapter.

Chapter 4, Understanding Speech Recogniser Output. This chapter investigates how SLU can be configured to understand the output of a speech recogniser, and to give distributions over dialogue acts that are useful for statistical dialogue systems. The main contribution is a robust method for SLU called the *CNet decoder*. The method is based on features

extracted from the full posterior distribution of recognition hypotheses encoded in the form of *word confusion networks*. Performance is evaluated on both in an off-line corpus-based evaluation, and on-line in a live user trial. It is shown that a statistical discriminative approach to SLU operating on the full posterior Automatic Speech Recognition (ASR) output distribution can substantially improve performance in terms of accuracy and overall *dialogue reward*. Furthermore, additional gains can be obtained by incorporating features from the previous system output.

Chapter 5, Dialogue State Tracking. This chapter presents a selection of methods for DST, including baseline approaches, the previous state of the art, and methods proposed recently by other research groups. The main contribution of this chapter is a discriminative method for DST using Recurrent Neural Networks (RNNs). This is applied directly to the speech recognition results to perform DST without the need for an explicit semantic decoder.

Chapter 6, Evaluation of Dialogue State Tracking. The Dialog State Tracking Challenges (DSTCs) are introduced, which were blind evaluations of DST methods using standardised datasets and evaluation metrics. These are used to evaluate the proposed methods for DST relative to a large selection of competing techniques. Discriminative DST using RNNs, including *word-based* trackers (which do not use any explicit SLU), are also evaluated in a live user trial.

Chapter 7, Unsupervised Learning for RNN Dialogue State Tracking. This chapter presents a method for improving DST by exploiting unlabelled dialogue data. This can be used to improve the parameters of the RNNs in adaptation to tracking new slots. This is evaluated using data from the third DSTC.

Chapter 8, Structured Output for RNN Dialogue State Tracking. This chapter presents a method for combining multiple RNNs trained on sub-components of the dialogue state, so that they output structured joint predictions. This is evaluated using data in a tourist information domain, where certain correlations and dependencies between slots exist.

Finally, chapter 9 concludes the thesis, giving a critical analysis of the benefits and limitations of the proposed approaches for SLU and DST. Some discussion is given to how these approaches may help in creating systems that can be used in expanding domains, so that they may understand and talk about an expanding scope of topics.

CHAPTER 2

OVERVIEW OF SPOKEN DIALOGUE SYSTEMS

This chapter gives an overview of Spoken Dialogue System (SDS) theory. Figure 2.1 presents the components of a typical SDS arranged in a pipeline. There is no consensus in the literature on the architecture of a dialogue system, but this is a representative reference (Pieraccini and Huerta, 2005). One cycle through this pipeline is one dialogue *turn*, i.e. one utterance from each participant, the user and the system. The following sections look at each of these components. An alternative overview is given in e.g. Jurafsky and Martin (2008).

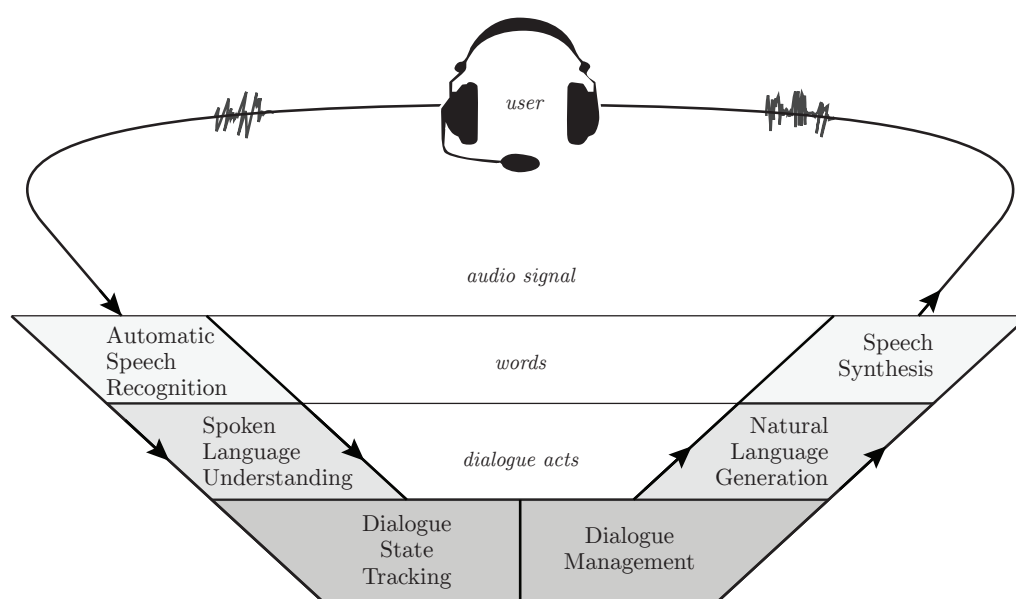


Figure 2.1: The pipeline architecture of a typical dialogue system.

This thesis focusses on task-oriented dialogue systems, where the domain can be represented using a specification of *slots*. Slots are variables which the user can either specify or ask about in the domain. For example, in a flight ticket buying system the user might specify destination and departure city slots, and ask for the value of the ticket price slot. Appendix A discusses slot-based dialogue domains further.

2.1 Automatic Speech Recognition

Automatic Speech Recognition (ASR) is the first component in the pipeline. This thesis is concerned with the steps in the pipeline that use the output of the Automatic Speech Recognition (ASR), specifically Spoken Language Understanding (SLU) and Dialogue State Tracking (DST).

Many state of the art systems for ASR use Hidden Markov Models (HMMs) to model the acoustics of speech (a review is given in e.g. Gales and Young (2007)). In recent years *discriminative* neural networks have been used to estimate the HMM state probabilities (Bourlard and Morgan, 1993; Deng et al., 2013; Hinton et al., 2012), and there is even work on doing end-to-end ASR using Recurrent Neural Networks (RNNs) (Graves and Jaitly, 2014; Robinson et al., 1996).

The form of the output of an ASR component is particularly important for the research on SLU presented in chapter 3. An ASR component essentially assigns a posterior probability to the words of an utterance given its acoustics. A typical form of output expected would be an *N-best list* of hypotheses with corresponding probabilities. This approximates the full distribution over all sentences with just the top *N* most probable, which is a limited approximation. Typically the variations in the top hypotheses involve mostly minor differences in articles and other short function words, with the result that many of the top hypotheses have the same meaning. Furthermore, words with low probability are likely to be omitted from the *N*-best list altogether. See the example in figure 2.2.

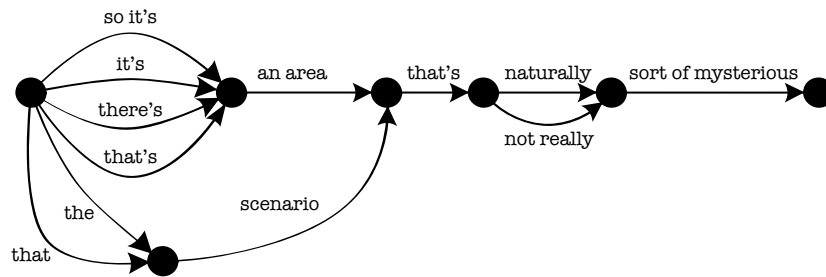
Intuitively a dialogue system should be able to benefit from all the information it can get from the ASR about the posterior distribution over words. Word lattices and word confusion networks provide a more informative summary of the distribution, without pruning the lower scoring words as in the *N*-best list. A word lattice is a directed graph that efficiently encodes possible word sequences, as shown in figure 2.2 (Murveit et al., 1993). Possible sentences are possible paths from the start node to the end node. Not depicted in the figure are the probability weightings on the edges, which allow for calculation of the probability of a path. Start and end time information for each word is also usually included.

Word confusion networks, like word lattices, are directed graphs that enumerate pos-

N-best list

Rank	Hypothesis	Log probability
1	it's an area that's naturally sort of mysterious	-7497.28
2	that's an area that's naturally sort of mysterious	-7508.93
3	it's an area that's not really sort of mysterious	-7505.33
4	that scenario that's naturally sort of mysterious	-7520.39
5	there's an area that's naturally sort of mysterious	-7518.45

Word lattice



Word confusion network

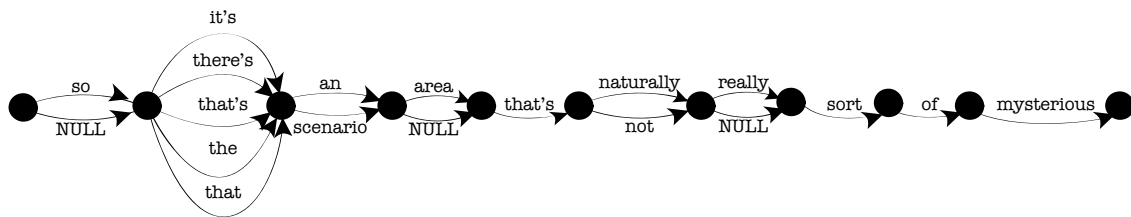


Figure 2.2: *N*-best lists, word lattices and word confusion networks are all structures that summarise the posterior distribution over sentences found by the ASR component. These examples are from Jurafsky and Martin (2008).

sible paths; however, they are more restricted in their structure (Mangu et al., 2000). As shown in figure 2.2, word confusion networks consist of a linear series of nodes representing word boundaries, with a set of mutually exclusive word hypotheses connecting each pair of consecutive nodes. Note that extra paths are added in the creation of the confusion network from the word lattice in figure 2.2, e.g. “*the scenario area*” is a possible path in the word confusion network but not the word lattice. However, every path that exists in the word lattice also exists in the word confusion network. The creation of a word confusion network may require arcs labelled with *NULL*, which are used to denote transitions that do not correspond to any word.

The restricted structure of the word confusion network allows for efficient calculations of quantities like the posterior probability of a given word sequence. Say there are nodes n_0, \dots, n_T connected by sets of edges W_0, \dots, W_{T-1} . Write $p(w)$ for the probability weight of a given edge, i.e. word $w \in W_i$ for some i . Then an ASR hypothesis is a choice from each set of edges, w_0, \dots, w_{T-1} with $w_i \in W_i$ with posterior probability:

$$P(w_0, \dots, w_{T-1}) = \frac{\prod_{i=0}^{T-1} p(w_i)}{\prod_{i=0}^{T-1} \sum_{w' \in W_i} p(w')} \quad (2.1)$$

where the denominator is the sum over all possible paths, and is equal to 1 if the $p(w)$ are normalised over each edge set, i.e. $\sum_{w \in W_i} p(w) = 1$. The probability of any subsequence can be similarly calculated. In section 4.1.2, such word sequence probabilities are used to summarise the whole posterior distribution for statistical SLU.

2.2 Spoken Language Understanding

Spoken Language Understanding (SLU), the next step in the pipeline after ASR, is the task of discerning the semantics of an utterance given the output of the ASR. Many spoken dialogue systems use a shallow level of semantics called dialogue acts, akin to the concept of a speech act (Searle, 1970). Dialogue act taxonomies are designed to capture just enough meaning in an utterance to facilitate a dialogue (de Mori, 2007). An SLU component, or semantic *decoder*, takes a sentence as input and gives a dialogue act as output.

Appendix B summarises the dialogue act taxonomy used in this thesis. Dialogue acts consist of a dialogue act type, which defines the general action of the utterance (e.g. requesting information, informing constraints, saying good-bye..), as well as a set of slot-value bindings that specify arguments of the act. For example *inform(food=chinese)* is a dialogue act of type *inform*, where the user is informing the system that they would like to constrain their search to venues serving Chinese food. This might be realised in English as

“A Chinese place please” or “Somewhere serving Chinese food.”

Many spoken dialogue systems use semantic template grammars to extract semantic concepts and discern the dialogue act (Ward, 1994; Young and Proctor, 1989; Zue et al., 2000). These generate parse trees for the sentence using a set of hand-written rules that map words and phrases to semantic concepts, and can group semantic concepts together to form sentences. Though often an effective technique, the rules are domain specific and require multiple iterations of user-testing before achieving adequate coverage (Young, 2002).

More complex grammar formalisms, such as Combinatory Categorical Grammars (CCGs), are able to model a wide range of complex linguistic phenomena (Steedman, 2000). These have been applied to SLU, where the grammars are relaxed so they may learn to parse ungrammatical spontaneous speech and erroneous speech recognition output (Kwiatkowski et al., 2011; Nguyen et al., 2006; Zettlemoyer and Collins, 2007). Other models that internally parse the input sentence include generative probabilistic models (He and Young, 2006; Miller et al., 1996) and those based on inductive logic programming (Zelle and Mooney, 1996). In contrast to techniques that represent the input sentence as a set of short phrases, or that use a finite window of context in a sentence, these parsing-based methods can readily model long range dependencies. This is important when applied to complex language, where concepts may be split up in the sentence, e.g. by relative clauses.

In some spoken dialogue domains, the possible dialogue acts may be enumerated to a small enough list to enable treating the SLU task as multi-class classification, where a single multi-class classifier is trained (Gupta et al., 2006; Schapire et al., 2005). This thesis however focusses on SLU tasks where the target annotation has some structure. While a single classifier would have to enumerate all the possible structures, causing sparsity in training examples, the following approaches attempt to exploit the structure in various ways.

One major difference among methods for SLU is whether they internally label sentences sequentially at the word level, or label the entire sentence. Methods that do sequential labelling often require aligned data, while methods that label the entire sentence can use unaligned data. Aligned labelling provides an alignment between the words in the input utterance and the target semantics, while unaligned labelling does not. Figure 2.3 gives an example to illustrate the difference between aligned and unaligned training data, and introduces *BIO tags* for aligned labelling. BIO tags provide a method of aligning spans of a sequence with labels, for example identifying the third to fifth words of “*I want Cocum Indian restaurant.*” as corresponding to a restaurant name ‘*Cocum Indian restaurant*’. This is clarified in the example.

Methods for SLU can generally be further split into two categories; *generative* and *discriminative* models. Generative models learn a joint probability distribution $P(\mathbf{x}, \mathbf{y})$ between

Aligned Labelling					
show	flights	going	from	Boston	to
B-action	I-action	O	B-from-city	I-from-city	B-to-city
New	York	today			
I-to-city	I-to-city	B-date			

Unaligned Labelling

show flights going from Boston to New York today
action=show flights; date = today; from city = Boston; to city = New York

Figure 2.3: An example of aligned and unaligned labels in the Air Travel Information System (ATIS) dataset (Dahl et al., 1994). BIO tags label the Beginning, Inside and Outside of sequential labels, allowing for labels spanning multiple words.

the input \mathbf{x} and the labels \mathbf{y} . The conditional distribution $P(\mathbf{y} | \mathbf{x})$ is then calculated using Bayes' rule, and used to assign labels to input examples. Discriminative methods directly model the conditional distribution $P(\mathbf{y} | \mathbf{x})$. This definition of *discriminative* should not be confused with *discriminative training* of generative models. Discriminative training of generative models is a technique for optimising parameters of an underlying generative model, while here discriminative is a property of the underlying probabilistic model.

Generative *dynamic Bayesian networks* can be defined to model the semantics of an utterance as a hidden structure jointly with the observed words (Acero and Wang, 2005; He and Young, 2006; Levin and Pieraccini, 1995; Schwartz et al., 1996). The Markovian assumption presents a challenge for modelling long-range dependencies between words, which can be solved by using a hierarchical hidden state (He and Young, 2006; Miller et al., 1996) – these models essentially parse the input sentence into a parse tree. A robust generative probabilistic grammar can also be learnt from data (Zettlemoyer and Collins, 2007). Machine translation has also been adapted to the task of learning SLU (Ramaswamy and Kleindienst, 2000), as has the formalism of *transformation rules* (Kate et al., 2005).

Discriminative models directly learn the conditional probabilities of the labels given a feature representation of the input utterance. Unlike generative models, such models do not make independence assumptions over the feature set. It is therefore easy to include arbitrary potentially useful features. Studies have shown that for this reason discriminative models can significantly outperform generative models in SLU tasks (Wang and Acero, 2006).

Discriminatively trained Markov Logic Networks have been applied to SLU with some success (Meza-Ruiz et al., 2008). Support Vector Machines (SVMs) have been used as discriminative classifiers for classifying syntactic tree features and semantic production rules (Kate and Mooney, 2006; Pradhan et al., 2004). Mairesse et al. presented the Semantic Tu-

ple Classifier (STC), which uses a set of discriminative SVM classifiers trained on unaligned data (Mairesse et al., 2009). This is presented in more detail in chapter 3.

Using a Conditional Random Field (CRF) (Lafferty et al., 2001) to do sequential labelling of input sentences is a particularly popular discriminative technique (Wang and Acero, 2006; Zhou and He, 2011) for SLU. The structure of the CRF can be adapted to jointly classify the topic of an utterance, using a *triangular* CRF (Jeong and Geunbae Lee, 2008). CRFs have also been used to create alignments from the word confusion network (Tur et al., 2013). Decoding with CRFs is described in more detail in chapter 3.

Recently there has been some work in applying neural network techniques to SLU. Convolutional neural networks, recurrent neural networks and long short-term memory recurrent neural networks have all been studied (Xu and Sarikaya, 2013; Yao et al., 2013, 2014). Similar to decoding with CRFs, these models treat SLU as a sequential labelling task.

2.3 Dialogue State Tracking

The term *dialogue state* loosely denotes a full representation of what the user wants at any point from the dialogue system. The dialogue state comprises all that is used when the system makes its decision about what to say next. For example, in a slot-based dialogue system the dialogue state includes the list of constraints the user has given so far in the dialogue.

An effective SDS must include a method of Dialogue State Tracking (DST) capable of accurately accumulating evidence over the sequence of a dialogue, and adjusting its prediction of the dialogue state according to the observations and context. A distribution over dialogue states is sometimes referred to as the *belief state* in the literature (Rapaport, 1986).

There has been an increase of interest in DST following the three Dialog State Tracking Challenges (DSTCs) (Henderson et al., 2014b,d; Williams et al., 2013), with a substantial number of papers published on the subject in the past two years. Many of the techniques mentioned here were evaluated in the Dialog State Tracking Challenges (DSTCs) alongside the methods proposed in chapter 5, and so appear in the results of the evaluation in chapter 6.

Similar to SLU, one key distinction in characterising techniques for DST is whether the model used is generative or discriminative. Generative models for DST jointly model both the inputs (typically the SLU hypotheses) and the dialogue state. Discriminative models directly capture the conditional probability over dialogue states, having observed the dialogue up to a certain point.

Generative approaches for DST are generally derived from the *Hidden Information State* model (Young et al., 2010), and use a dynamic Bayesian network to model observations

from the SLU jointly with the dialogue state. A set of hidden random variables represents the dialogue state, and the whole model is referred to as the *user model*. Chapter 5 describes a representative member of this class of models, the Bayesian Update of Dialogue State (BUDS) system (Thomson and Young, 2010). Other such methods employing a dynamic Bayesian network include Kadlec et al. (2014a,b); Williams (2010). Typically these use parameters optimised by the system designer, though there are techniques to exploit dialogue data to optimise the parameters (Lee et al., 2014; Thomson et al., 2010a). Another proposition is to use a secondary step to post-process the output of a generative model (Kim et al., 2013).

Rule-based approaches have also been proposed for the task of DST, which use a small set of hand-crafted rules to update the dialogue state given observations from the SLU (Smith, 2014; Wang and Lemon, 2013). The *focus* baseline tracker presented in chapter 5 is a simple example. Sun et al. (2014a) generalise this class of models and present a method for learning sets of rules.

Discriminative approaches to DST are generally *machine learnt*, i.e. their parameters are optimised using labelled dialogue data. A more detailed description of discriminative methods for DST is presented in chapter 5, but a summary is given below.

An early discriminative method is presented in Bohus and Rudnicky (2006), which uses a Maximum Entropy linear model. In work following the start of the DSTCs, Maximum Entropy and neural network classifiers have been popular models (Henderson et al., 2013; Lee and Eskenazi, 2013; Metallinou et al., 2013; Ren et al., 2014b,a; Sun et al., 2014b; Williams, 2012, 2013).

Top accuracies in the second DSTC were obtained using a discriminative ranking model, similar to that employed for competitive ranking of web documents in document retrieval and search (Williams, 2014).

All of the discriminative methods mentioned so far make use of a static classifier that is activated at each turn in the dialogue. The probability over dialogue states in a given turn is conditioned on a feature representation of the whole dialogue history up to that point. In order to classify arbitrary length sequences of dialogue states, which may change from turn to turn, special feature vectors must be designed to capture the sequence while remaining of a fixed dimension. For example Henderson et al. (2013) use a sliding window and accumulation of features over turns. Metallinou et al. (2013) use a set of *history* features that e.g. give the sum and average of stationary features of the SLU, or count specific events in the SLU history (such as the number of times a hypothesis has appeared at a certain rank).

Other discriminative methods directly model the sequential nature of the inputs. Conditional Random Fields have been used to model arbitrary length dialogues, with a fixed num-

ber of features for every turn in the dialogue (Kim and Banchs, 2014; Lee, 2013). However these models must use features with a finite number of possible values, and so continuous features must be quantised into discrete features.

2.4 Dialogue Management

Once the dialogue state has been estimated by the DST component, the SDS must decide what to say next. This process is called dialogue management. This thesis focusses on estimating the input to the dialogue manager rather than the dialogue management itself. However, when the proposed approaches to SLU and DST are evaluated in a live user trial, dialogue management is very important. In particular, the live evaluation of chapter 5 uses online Gaussian process reinforcement learning. This section gives a brief summary of approaches to dialogue management.

Conventional dialogue systems typically maintain one hypothesis for the dialogue state, effectively making the assumption that the state is known. Various frameworks have been proposed to interact with this state and select the next system action, for example approaches based on flow-charts (Lucas, 2000; McTear, 1998; Sutton et al., 1996), form filling systems (Goddeau et al., 1996), and systems that use logical inference and planning (Larsson and Traum, 2000).

None of these approaches suggests a way of learning which actions should be taken, leaving this to be hand-crafted by the system designer. Casting the problem as a Markov Decision Process (MDP) allows learning of the action selection model (Biermann and Long, 1996; Levin and Pieraccini, 1997; Singh et al., 1999). Though the single hypothesis for the state can be augmented to include details of the uncertainty (Bohus and Rudnicky, 2005), an approach that is more theoretically well-founded is to instead use a Partially Observable Markov Decision Process (POMDP).

2.4.1 Statistical Dialogue Systems

The statistical approach to dialogue management maintains a distribution over dialogue states (Young, 2002). By explicitly modelling the uncertainty in dialogue, statistical systems can better deal with the ambiguities inherent in natural language and arising from uncertain speech recognition (Thomson, 2009).

In the statistical framework, the ASR is configured to output a distribution over sentences using e.g. an N -best list, or word confusion network (figure 2.2). The SLU then uses this

distribution to output an M -best list¹ of dialogue acts (methods for doing this are discussed in chapter 4). The DST then uses the M -best list to update the distribution over the dialogue state.

Dialogue management can then be cast as a POMDP, which considers the distribution over the dialogue state when selecting the next action (Roy et al., 2000; Williams and Young, 2005). The POMDP-based BUDS framework presented in Thomson and Young (2010) gives a baseline statistical system, which is used in the evaluations of this thesis.

In a POMDP, the *policy* π makes the selection of which action to take given a distribution over dialogue states $P(s)$. Writing $b = P(s)$ (for the belief state), the policy is a function $\pi(b)$ that selects the next user action. Learning of the policy is facilitated by defining a *reward* $r(s, a)$ for every state s and action a . The total reward over a dialogue is therefore:

$$R = \sum_{t=0}^{T-1} r(s_t, a_t) \quad (2.2)$$

where T is the number of turns in the dialogue. The policy is learnt by maximising the expected reward $\mathbb{E}(R)$. Note that the states s_t are not exactly known, so the $P(s_t)$ estimations from the DST are used in this expectation. This type of learning is called reinforcement learning, as good decisions in learning are reinforced by good rewards. An important entity in reinforcement learning is the $Q(s, a)$ function, defined as the expected future reward for taking action a in state s .

Gaussian processes can be used to model the Q function in Gaussian process reinforcement learning. This technique has proven efficient enough to use for learning a policy in direct interaction with users (Gašić et al., 2010, 2013), while previous approaches have relied heavily on simulated interactions. In the live trial of section 6.5, reinforcement learning using Gaussian processes is used to quickly learn a policy for dialogue managers in a variety of conditions.

2.5 Natural Language Generation and Speech Synthesis

Generation of the text corresponding to a system’s chosen dialogue act (Natural Language Generation (NLG)), and then synthesising this text into audio (Speech Synthesis (SS)) are both steps that are not studied in detail in this work. Nevertheless they are essential for a complete dialogue system, and so this section briefly describes common approaches found in the literature.

¹ M is chosen when referring to SLU output throughout this thesis to avoid confusion with the N -best output of the ASR.

In the literature on NLG for dialogue systems, there are a variety of proposed approaches falling roughly into three categories. The first category is simple template-based generation. This method is used for the studies in this work, and operates by using a fixed set of simple hand-written templates. For example, the template:

inform(name=x, food=y) → “x is a nice place serving y food”

could generate sentences like “*Cocum is a nice place serving Indian food.*”

Learnable techniques for NLG also exist, constituting the second category of methods. Supervised learning approaches allow for adapting the language generated according to sets of example outputs (Stent et al., 2004; Walker et al., 2001). The NLG problem has also been cast as a Markov Decision Process (MDP) so that the model may be learnt using reinforcement learning (Rieser and Lemon, 2011, 2010). Another technique treats generation in dialogue systems as a search for the most likely sequence of semantic concepts and words using *Factored Language Models* (Mairesse and Young, 2014).

Lastly, the third category of methods employs *conventional* NLG, as it is developed in the text generation literature (Reiter and Dale, 2000). This is a pipeline architecture that builds up sentence plans to be mapped to text realisations. Such methods have been used to tailor output to users (Moore et al., 2004) and the dialogue context (Isard et al., 2003).

In the next step, termed Speech Synthesis (SS), the system’s chosen text is converted into speech. Unit selection is used in many systems, constructing a waveform by concatenating segments of recordings held in a database. This is implemented in e.g. the Festival synthesiser (Taylor et al., 1998).

Another widely used approach, used in this work, is HMM-based SS. This is a statistical method using a generative model of speech (Zen et al., 2007). These models can learn dialogue context sensitive behaviours, giving more natural speech (Tsiakoulis et al., 2014).

Neural networks have also been applied with success to the problem of SS (Zen et al., 2013). Most recently, researchers have found improvements in using RNNs to model the generation of speech from text (Zen et al., 2014).

CHAPTER 3

SPOKEN LANGUAGE UNDERSTANDING

Spoken Language Understanding (SLU) is the process of converting spoken words into a semantic representation more readily processed by the next components in the pipeline of a dialogue system. An SLU component must be able to deal with a variety of natural language expressions and distill these into the important details, such as which slot-value constraints the user is giving in a slot-based dialogue system. This is a core capability of most spoken dialogue systems.

The SLU discussed in this chapter outputs dialogue acts in the format described in appendix B. For example a sentence such as “*What is the number of a Chinese place*” must be mapped to *request(phone, food=chinese)* by the SLU.

This chapter presents three methods for SLU in a slot-based dialogue system, contrasting a grammar-based approach with two *discriminative* statistical models. One discriminative method employs a Conditional Random Field (CRF) to do sequential labelling, and requires training data where words are aligned to labels. The other discriminative method, the Semantic Tuple Classifier (STC), does not treat the input as a sequence and does not require labelled alignments.

A dataset for SLU is presented, composed of dialogues with a restaurant information dialogue system in noisy conditions (in an automobile). The dialogues are labelled with the correct dialogue acts, including alignments from the words to the sub-components of the dialogue acts. This dataset allows a side-by-side evaluation of these three SLU methods, and is then used in the following chapter for further analysis.

3.1 Methods for Spoken Language Understanding

Section 2.2 gave a review of methods used for SLU in spoken dialogue systems. This section will present in more detail three methods, which are evaluated on a common set of dialogue data. Details of a grammar-based *decoder* are given in section 3.1.1, which is the previous state of the art used in research by the Cambridge dialogue group. Section 3.1.2 presents a CRF model for solving SLU as a sequential labelling task. This approach is widely adopted in the literature. This gives a contrast for the approach in section 3.1.3, which uses discriminative classifiers trained on unaligned data.

3.1.1 Using a Robust Grammar

Phoenix is a robust semantic decoder that uses manually constructed grammars designed to detect keywords and phrases, and converts them into semantic tags (Ward, 1994). The Phoenix parser is designed to be robust to word errors and searches to find the best possible parse of the input according to a metric that seeks to find the longest spanning phrases. The grammar used in this evaluation has been specially written for the restaurant information domain over a period of several years, and has been refined to work effectively on the Automatic Speech Recognition (ASR) hypotheses generated by the CUED Dialogue Systems group's speech recogniser¹.

For decoding in a dialogue system, the Phoenix decoder runs on each of the top ASR hypotheses in the *N-best list*, producing a single dialogue act for each. Each dialogue act is weighted by the corresponding posterior probability assigned by the ASR in the *N-best list*. Duplicates are merged to give the final distribution over dialogue acts. This configuration is termed *N-best semantic decoding*.

3.1.2 Using a Conditional Random Field

Semantic decoding may be considered a sequential labelling task (Wang et al., 2005a). In this formulation, the words in the input sentence must be labelled with *BIO tags*, which encode the slot bindings in the dialogue act. Figure 3.1 gives examples of BIO tag encodings in the restaurant information domain and how these relate to the Cambridge dialogue act format (appendix B). To recover the full dialogue act it is also necessary to identify the dialogue act type.

¹An HTK-based recognition system, with a vocabulary of around 3000 words and a trigram language model (Young et al., 2006)

Words:	okay	what	is	the	price	range
Labels:	O	O	O	O	B-unbound-slot	I-unbound-slot
Act type:	<i>request</i>					
Dialogue act:	<i>request(pricerange)</i>					
Words:	cheap	north	american	food		
Labels:	B-pricerange	B-food	I-food	O		
Act type:	<i>inform</i>					
Dialogue act:	<i>inform(pricerange=cheap, food=“North American”)</i>					
Words:	where	is	the	Chinese	place	
Labels:	B-unbound-slot	I-unbound-slot	O	B-food	O	
Act type:	<i>request</i>					
Dialogue act:	<i>request(area, food=Chinese)</i>					
Words:	thank	you	good	bye		
Labels:	O	O	O	O		
Act type:	<i>bye</i>					
Dialogue act:	<i>bye()</i>					

Figure 3.1: Example BIO tag encodings in the restaurant information domain (see appendix A). Values in the sequence of words are labelled with the slots they correspond to. Unbound slots (slots that are being requested) are labelled with a special *unbound-slot* category. The dialogue act format can be recovered from the alignment and the act type. Note in the third example it is necessary to resolve ‘*where is*’ to the *area* slot.

Linear chain CRFs (Lafferty et al., 2001) have been successfully applied to the sequence labelling task (Raymond and Riccardi, 2007; Wang and Acero, 2006; Wang et al., 2005b). This remains a popular method, used in recent studies e.g. Li et al. (2014). Other models have since been proposed for performing sequence labelling, such as Recurrent Neural Networks (RNNs) (Yao et al., 2013, 2014). The CRF method is presented here as a representative approach to SLU that treats the problem as sequence labelling. This is compared with a method using binary discriminative classifiers, which take the entire sequence as input to solve the same problem (presented in the next section).

A CRF is used to obtain the most likely sequence of BIO tag labels \hat{Y} given the input word sequence:

$$\hat{Y} = \arg \max_Y P(Y | X) \quad (3.1)$$

where $X = (x_0, \dots, x_{T-1})$ is the input word sequence and $Y = (y_0, \dots, y_{T-1})$ is a sequence of labels. The joint distribution of Y and X takes the form of a linear chain with first order Markov property, and the conditional probability is given as:

$$P(Y | X) = \frac{1}{Z(X)} \exp \left(\sum_{k,t} \lambda_k f_k(y_{t-1}, y_t, X) \right) \quad (3.2)$$

where the f_k are feature functions and λ_k are learnt weights of the model. $Z(X)$ is a normalisation term. The weights can be learnt by e.g. using a gradient descent algorithm. The Viterbi algorithm can then be used to find the optimal sequence of labels \hat{Y} .

Note that as the conditional distribution is being learnt directly, rather than the joint distribution, this is a discriminative model. Arbitrary and potentially correlated features f_k can be used in the model, for example those that use the identity of neighbouring words. Wang and Acero (2006) found that for this reason CRFs are able to outperform *generative* models, which model the joint distribution $P(X, Y)$. Such generative models were the previous state of the art (Wang et al., 2005a). An example of such generative models is the Hidden Markov Model (HMM):

$$P(X, Y) = \sum_{t=0}^{T-1} P(y_t | y_{t-1}) P(x_t | y_t) \quad (3.3)$$

Following Jeong and Geunbae Lee (2008), the feature functions used in this work are binary indicator functions that output either 1 or 0:

- **Transition features** for every pair of labels y'_{t-1} and y'_t , a binary indicator function

$$f_k(y_{t-1}, y_t) = \begin{cases} 1 & \text{if } y_{t-1} = y'_{t-1} \text{ and } y_t = y'_t \\ 0 & \text{otherwise} \end{cases} \quad (3.4)$$

- **Neighbouring word features** for every label, word pair y'_t, x'_t and every $\tau \in -2, -1, 0, 1, 2$, a binary indicator function

$$f_{k,\tau}(y_t, X) = \begin{cases} 1 & \text{if } y_t = y'_t \text{ and } x_{t+\tau} = x'_t \\ 0 & \text{otherwise} \end{cases} \quad (3.5)$$

note that at $\tau = 0$ this gives the identity of the current word.

A linear chain CRF such as described above can be used to tag the word sequence, but it is also necessary to classify the dialogue act type in order to recover the whole dialogue act. This can be done with a complementary model in multiple configurations, but using an altered CRF structure, the Triangular CRF, to jointly classify the dialogue act type and label the sequence is found to perform best (Jeong and Geunbae Lee, 2008).

The triangular CRF model introduces another variable z , the *topic*. In this context z is the dialogue act type. The optimal sequence \hat{Y} and topic (dialogue act type) \hat{z} are jointly selected:

$$\hat{Y}, \hat{z} = \arg \max_{Y, z} P(Y, z | X) \quad (3.6)$$

the full dialogue act can be recovered from \hat{Y} and \hat{z} .

The conditional probability in a triangular CRF is of the form:

$$P(Y, z | X) = \frac{1}{Z(X)} \exp \left(\sum_{k,t} \lambda_k f_k(z, y_{t-1}, y_t, x_t) \right) \exp \left(\sum_k \lambda_k f_k(z, X) \right) \quad (3.7)$$

i.e. the factors that are functions of Y and X are allowed to further depend on z , and the last term introduces a series of factors depending on X and z . The $f_k(z, X)$ feature functions are binary indicator features for every word in X and possible dialogue act type z . The $f_k(z, y_{t-1}, y_t, x_t)$ features are the same binary indicator features as above, but expanded to also depend on z – i.e. for every possible dialogue act z' and feature function f_k from the

linear CRF there is a new feature function:

$$f_k(z, y_{t-1}, y_t, x_t) = \begin{cases} f_k(y_{t-1}, y_t, x_t) & \text{if } z = z' \\ 0 & \text{otherwise} \end{cases} \quad (3.8)$$

The parameters of the model $\{\lambda_k\}$ are learnt as in Jeong and Geunbae Lee (2008)². The TriCRF³ implementation of triangular CRFs is used, with all the recommended configurations.

3.1.3 Using Discriminative Classifiers

This section presents the Semantic Tuple Classifier (STC) decoder, which uses a set of discriminative classifiers to discern the dialogue act from the n -gram counts in the input sentence (Mairesse et al., 2009).

Support Vector Machines (SVMs) are used for classification, with linear kernels, and outputs are converted to probabilities using a sigmoid function whose shape is learnt from cross-validation (Platt, 1999).

The STC decoder requires a set of classifiers to be trained. One multi-class classifier is used to predict the dialogue act type, and a binary classifier is used to predict the existence of each possible slot-binding. The training data for these classifiers does not need to be aligned, as with the CRF approach of the previous section.

The classifiers take an utterance u as input, and give the probability of each dialogue act type, $P(d\text{-type}_i | u)$ as well as the probability of each possible slot binding x , $P(x|u)$. The probability of a dialogue act D with type $d\text{-type}$ and slot-bindings X is approximated by:

$$P(D | u) = P(d\text{-type} | u) \prod_{x \in X} P(x | u) \prod_{x \notin X} (1 - P(x | u)) \quad (3.9)$$

The vector representation of an utterance u as presented in (Mairesse et al., 2009) is $f_i = C_u(n\text{-gram}_i)$ where $C_u(ng)$ counts how many times the n -gram ng occurs in utterance u . n is allowed to range from 1 to 3, i.e. words and sequences of two and three words are counted (unigrams, bigrams, and trigrams). Only a small number of n -grams present in the training data occur in any single utterance, meaning this is a sparse representation. This allows for fast training and classification with Support Vector Machines (SVMs).

²The parameters are first initialised by individually optimising $P(Y | z, X)$ and $P(z | Y, X)$, before going on to optimise $P(Y | z, X)$. Optimisation is performed using the limited memory Broyden–Fletcher–Goldfarb–Shanno algorithm.

³Available at <http://github.com/minwoo/TriCRF>

The informable slots in the domain S_{inf} , those which the user may give constraints on (see appendix A), are split into two categories – *enumerable* and *non-enumerable* slots. Non-enumerable slots are those with a large number of possible values, where there may be certain slot bindings with very few or no examples in the training data. For an enumerable slot s , a classifier is learnt for the slot-bindings $s = v$ and $s \neq v$ for every $v \in V_s$. For non-enumerable slots, occurrences of values in the input string are replaced with generic tags, and a generic classifier is learnt for $s = \langle \text{tagged-}s\text{-value} \rangle$ and $s \neq \langle \text{tagged-}s\text{-value} \rangle$.

In the restaurant information domain, the *area* and *pricerange* slots are considered enumerable while the *food* and *name* slots are non-enumerable. There are 91 possible values for food, and 113 for name, so learning a classifier for each value would be infeasible with realistic amounts of training data. For example, consider the input string:

“Does Seven Days serve Chinese food?”

The instances of possible values for *name* and *food* are replaced with generic tags:

“Does $\langle \text{tagged-name-value} \rangle$ serve $\langle \text{tagged-food-value} \rangle$ food?”

This then becomes a positive instance in training for the $\text{food} = \text{tagged-food-value}$ and $\text{name} = \text{tagged-name-value}$ classifiers (as well as an instance of $d\text{-type} = \text{confirm}$). For non-enumerable slots, slot-bindings that are very frequent in the data and the *Dontcare* binding are still learnt individually.

Mairesse et al. evaluated the performance of the STC model on the Air Travel Information System (ATIS) dataset (Dahl et al., 1994). STC was found to outperform the state of the art in generative models (the Hidden Vector State model (He and Young, 2006)) and perform competitively with respect to the state of the art in grammar-based decoding (Zettlemoyer and Collins, 2007). No direct comparison was possible with CRF decoding, due to inconsistent training and testing splits used in the literature. Section 3.3 however presents an evaluation of the Phoenix grammar, triangular CRF decoding and STC decoding on a common set of dialogue data.

The STC decoder is extended in chapter 4 to better handle uncertainty in the ASR hypotheses.

3.2 In-Car Spoken Language Understanding Corpus

The methods for SLU described in the previous sections are evaluated in an off-line corpus-based evaluation. This section introduces the Cambridge In-Car SLU corpus, presented in Henderson et al. (2012) and made available for download online⁴. The corpus has since

⁴The corpus may be downloaded at <http://www.repository.cam.ac.uk/handle/1810/248271>

	Train set	Test set
Dialogues	1522	644
Utterances	10571	4882
Male : Female	28 : 31	15 : 15
Native : Non-Native	33 : 26	21 : 9

Table 3.1: Statistics of the in-car Spoken Language Understanding Corpus

been used by other research groups to evaluate techniques for dialogue act type classification (Chen et al., 2013a) and unsupervised slot induction (Chen et al., 2013b, 2014).

The dialogue corpus data for off-line evaluation was collected using a restaurant information system for the city of Cambridge. Users can specify restaurant suggestions by area, pricerange and food type and then can then query the system for additional restaurant specific information such as phone number, post code, signature dish and address.

To achieve a range of differing noise conditions, participants were asked to interact with different systems operating in a variety of conditions related to in-car dialogues; in a stationary car with the air conditioning fan on and off, in a moving car and in a car simulator (Gašić et al., 2012; Tsiakoulis et al., 2012). The average Word Error Rate (WER) obtained by the speech recognition in this corpus was 37.0%.

Each section of the corpus has an equal distribution of the different in-car conditions. Some basic statistics of the corpora are given in Table 3.1.

3.2.1 Labelling

Each utterance in the dialogue data is labelled with the correct dialogue act, as well as a BIO tag encoding aligning the words to slot-bindings.

To facilitate labelling of the dialogue acts, the raw audio was first transcribed using crowd-sourced workers. A Phoenix grammar was run on these transcriptions, to give fairly reliable initial dialogue act labels. These initial labels were then corrected by hand.

The alignments between slot-bindings and the word sequences (required for the CRF model) were then created automatically. To facilitate finding slot-bindings, a set of possible text realisations, or *aliases*, for each slot-binding was created. This set of aliases, for example says that *food* = “*north american*” can be expressed by the words *american* or *north american*. Full details of the aliases used can be found in appendix C.

Finding the alignments is not trivial. For sentences like “*where is it*” and “*how much is it*”, it is necessary to decide which words correspond to the *unbound-slot* category for the *area* and *pricerange* slots respectively. Figure 3.1 gives an example that uses the alias

‘*where is*’ to identify the *area* slot. The *Dontcare* values also need some consideration, for example aligning “*anywhere*” with *area = Dontcare*.

The set of aliases created for this domain allows for finding alignments from the transcriptions with all but 131 of the total 10,816 slot-bindings (98.8%) in the training data. When applied to the ASR hypotheses only 73.2% of alignments can be found, due to missing words.

For CRF decoding, the aliases must be used to resolve the alignment given on testing utterances with slot-bindings.

3.2.2 Metrics

For conventional non-statistical dialogue systems, an important metric to assess the quality of the output of a semantic decoder is the *F-score*, the harmonic mean of the precision and recall of true semantic items in the top semantic hypothesis.

Let the true reference dialogue act for an utterance be D_{ref} , given by the dialogue act type $\text{d-type}_{\text{ref}}$ and the set of slot bindings X_{ref} . Suppose the top scoring hypothesis output by the semantic decoder is D_{hyp_0} with dialogue act type $\text{d-type}_{\text{hyp}_0}$. Then the F-score of this output is:

$$F = \frac{2|A \cap B|}{|A| + |B|}$$

$$\text{where } A = (X_{\text{ref}} \cup \{\text{d-type}_{\text{ref}}\}) \quad (3.10)$$

$$\text{and } B = (X_{\text{hyp}_0} \cup \{\text{d-type}_{\text{hyp}_0}\})$$

As noted in section 2.3 , in a statistical dialogue system the distribution over all possible dialogue acts is used to update the distribution over dialogue states. It is therefore important that the distribution output by the SLU component accurately reflects the underlying uncertainty. The Item Cross Entropy (ICE) between the hypotheses and the true semantics measures the overall quality of a distribution and is shown to correlate strongly with the performance of a statistical Spoken Dialogue System (SDS) as measured by the *dialogue reward* (Thomson et al., 2008).

Suppose the full distribution output by a semantic decoder is a list of hypotheses D_{hyp_i} with corresponding probabilities p_i for $i = 0, \dots, M - 1$, such that $\sum_{i=0}^{M-1} p_i = 1$ and $p_0 \geq p_1 \geq p_2 \geq \dots \geq p_{M-1}$, where D_{hyp_i} is given by the dialogue act type $\text{d-type}_{\text{hyp}_i}$ and the set of slot bindings X_{hyp_i} . The ICE score is calculated from the confidences, $c(\cdot)$, of each dialogue

act type and slot binding as given by this M -best list:

$$c(\text{d-type}) = \sum_{i=0}^{M-1} \begin{cases} p_i & \text{if d-type} = \text{d-type}_{\text{hyp}_i} \\ 0 & \text{otherwise} \end{cases} \quad (3.11)$$

$$c(x) = \sum_{i=0}^{M-1} \begin{cases} p_i & \text{if } x \in X_{\text{hyp}_i} \\ 0 & \text{otherwise} \end{cases}$$

These are compared with the true distribution of the semantics, $c^*(\cdot)$,

$$c^*(\text{d-type}) = \begin{cases} 1 & \text{if d-type} = \text{d-type}_{\text{ref}} \\ 0 & \text{otherwise} \end{cases} \quad (3.12)$$

$$c^*(x) = \begin{cases} 1 & \text{if } x \in X_{\text{ref}} \\ 0 & \text{otherwise} \end{cases}$$

to give the ICE score:

$$\text{ICE} = \frac{1}{1 + |X_{\text{ref}}|} \sum_{y \in Y} \log(c(y)c^*(y) + (1 - c(y))(1 - c^*(y))) \quad (3.13)$$

where Y is a set containing all the possible dialogue act types and slot value pairs, and the arguments to the log function are floored to remain positive.

In evaluations, the training corpus is used to train a semantic decoder, then the whole test corpus is decoded. The F-score and ICE for each test utterance are then calculated and averaged.

3.3 Evaluation of Spoken Language Understanding Methods

The Phoenix grammar, triangular CRF decoder, and STC decoder were evaluated on the in-car SLU data. Table 3.2 presents the F-scores achieved by the decoders on the test set when trained and tested on the transcriptions (not using ASR). The three methods perform similarly, though the triangular CRF performs slightly worse than the STC and Phoenix grammar.

Performance on ASR output is of more interest when deploying a spoken dialogue system. Table 3.3 presents the results of evaluating the decoders on ASR output. Each decoder is configured to output a single dialogue act for each ASR hypothesis in the N -best list

Method	F-score
Phoenix grammar	0.954 ± 0.005
Triangular CRF	0.945 ± 0.005
STC	0.953 ± 0.005

Table 3.2: Training and testing on transcriptions. Results are written $\mu \pm 1.96\sigma$ where μ is the mean over all the test utterances and σ is the standard error in the estimation of the mean.

Method	Trained on	F-score	ICE
Phoenix grammar	–	0.694 ± 0.012	2.784 ± 0.116
Triangular CRF	transcriptions	0.704 ± 0.012	2.539 ± 0.104
	top hypotheses	0.674 ± 0.012	2.747 ± 0.112
Semantic Tuple Classifier	transcriptions	0.707 ± 0.012	2.621 ± 0.113
	top hypotheses	0.691 ± 0.012	2.561 ± 0.108

Table 3.3: Training either on transcriptions or the top ASR hypothesis, then testing on the top ASR hypothesis. Results are written $\mu \pm 1.96\sigma$ where μ is the mean over all the test utterances and σ is the standard error in the estimation of the mean.

($N = 10$). The STC decoder selects the dialogue act that maximises the probability given by equation (3.9). These are then weighted by the corresponding probabilities from the ASR and duplicates are merged to give an M -best list of dialogue acts ($M \leq N$).

The discriminative statistical methods, triangular CRF and STC decoders, can either be trained on the clean transcriptions or on noisy ASR output. Both configurations are evaluated here. In the case of training on ASR output, only the top ASR hypothesis from the N -best list is used.

Table 3.3 gives the F-scores and ICE scores for each system. All three systems perform similarly, with few contrasts being statistically significant. The notable exception is the triangular CRF decoder trained on the noisy ASR output. The triangular CRF performs better when trained on clean data and then applied to the noisy data. Recall only around 73% of alignments can be found between the slot-bindings and the noisy word sequences. The lack of training examples may hinder the performance of the CRF in this case. In terms of ICE, the STC decoder benefits from being trained on noisy data.

Other than the CRF trained on noisy data, all systems achieve a similar F-score, meaning that the top SLU hypotheses are of similar quality across systems. The ICE score shows more variability, with the discriminative statistical systems able to improve on the Phoenix grammar. This implies the statistical decoders are able to extract more useful information

when applied to the more noisy ASR hypotheses lower down the N -best list.

3.4 Conclusions

This chapter has presented three methods for SLU, comparing a non-statistical but robust grammar to two discriminative methods. The discriminative triangular CRF model treats the task as sequential labelling and outputs an alignment, while the STC model extracts features from the whole utterance and does not use any alignments.

These methods were evaluated side-by-side on a set of dialogue data, the in-car SLU corpus. The three methods performed similarly, though the statistical methods were able to extract more information from the ASR N -best list and the CRF model had difficulty training on the ASR output.

In the following chapter, the STC framework is chosen as a basis to develop a decoder that better exploits the information in the posterior distribution over words as given by the ASR. The STC decoder is chosen over a CRF approach for two main reasons:

1. The STC approach does not require aligned data. This means that arbitrary feature functions can be explored, which are not necessarily aligned with the labels in any way. As discussed in section 3.2.1, finding the alignments is not trivial and requires further work by an expert (such as creating a set of aliases). The results here suggest that well aligned training data is important for the CRF model, but it is particularly difficult to find alignments when using noisy inputs.
2. The STC model easily supports continuous features. CRFs as presented in the SLU literature are used with discrete features such as those used in this chapter. Continuous features may be converted into discrete features by quantisation, and CRFs may be adapted to deal with continuous features, but learning CRFs is best understood for discrete features (Feng et al., 2006). For example Tur et al. (2013) look at applying CRF decoding to *word confusion networks*, using discrete features that do not depend on the transition probabilities given by the network.

CHAPTER 4

UNDERSTANDING SPEECH RECOGNISER OUTPUT

A deployed dialogue system may be exposed to noisy conditions and a wide variety of speakers, adversely affecting the performance of the Automatic Speech Recognition (ASR). For example, in the in-car dialogues presented in section 3.2.1 the average Word Error Rate (WER) was 37.0%, meaning that more than one in every three words is expected to be misidentified by the recogniser. It is therefore important that the Spoken Language Understanding (SLU) is robust to this noise and fully exploits the output of the ASR.

A statistical dialogue system can exploit a distribution of hypotheses from an SLU component, given by an M -best list of the top M dialogue acts and associated probability scores.

The previous chapter introduced a method to allow any conventional semantic *decoder* to output a distribution over dialogue acts given an ASR N -best list. In N -best semantic decoding, each of the N ASR hypotheses is individually mapped to a single dialogue act, then duplicates are merged to give a M -best list of dialogue acts ($M \leq N$). This is a simple way of handling the distribution output over words by the ASR employed in several dialogue systems in the literature (He and Young, 2003; Thomson, 2009; Williams et al., 2010). One critical limitation of N -best semantic decoding is that unless very long N -best lists are used, the final pruned lists of semantic hypotheses are often very short¹ and the resulting approximation is quite poor. Secondly each element of the M -best list is being computed from a single element of the N -best word list even though elements of the list are highly correlated.

It is possible to use word-level confidences from the ASR *word confusion network* to improve the quality and robustness of outputs in related problems, such as call classification (Hazen et al., 2002; Tur et al., 2004, 2002) and spoken utterance retrieval (Saraclar, 2004).

¹Frequently less than 4 for the tourist information domain, and $N = 10$.

Joint models have been proposed for ASR and slot filling so that the two processes can be optimised together (Deoras et al., 2012, 2013). Alternatively, a Conditional Random Field (CRF) decoder can be adapted and applied to the word confusion network to provide improvements relative to N -best decoding (Tur et al., 2013). This chapter finds similar gains from exploiting the word confusion network for SLU in the Semantic Tuple Classifier (STC) framework.

The main contribution of this chapter is the *CNet decoder*, which extracts features from the full posterior distribution of recognition hypotheses, as encoded in the form of a word confusion network. Its performance is evaluated on both an off-line corpus and on-line in a live user trial. It is shown that this statistical *discriminative* approach to SLU operating on the full posterior ASR output distribution can substantially improve performance both in terms of accuracy and overall *dialogue reward*. Furthermore, additional gains can be obtained by incorporating features from the system’s output.

Williams (2014) and Sun et al. (2014b) reimplemented the CNet decoder using data from the second Dialog State Tracking Challenge (DSTC) (Henderson et al., 2014b), in the restaurant information domain (also used in this chapter for evaluation). They confirmed the findings of the following sections – that features derived from the word confusion network give improved performance over features derived from the N -best list for SLU tasks.

4.1 Semantic Decoder Configurations

Figure 4.1 illustrates three alternative methods for generating an M -best list of dialogue acts given the output of the ASR. N -best semantic decoding (figure 4.1a) was used in chapter 3 to convert the output of conventional semantic decoders, which output a single dialogue act for a given sentence, into an M -best list. This chapter investigates two alternatives in an attempt to extract higher quality SLU M -best lists- $N \times M$ -best semantic decoding and *one run* semantic decoding.

In $N \times M$ -best semantic decoding, a statistical semantic decoder is used that can output an M -best list of dialogue acts for a single input sentence, such as the STC decoder. An M -best list is generated for every hypothesis in the ASR N -best list, the probability scores are then weighted by the corresponding ASR probabilities, and the collection of N M -best lists are then merged. In contrast to N -best semantic decoding, M may be chosen freely and is not constrained above by N . It is expected that this method will output a more varied distribution by generating multiple SLU hypotheses per ASR hypothesis.

When training the STC model for use in $N \times M$ -best semantic decoding, a data point is created for each of the top k hypotheses of each training utterance with the same semantic

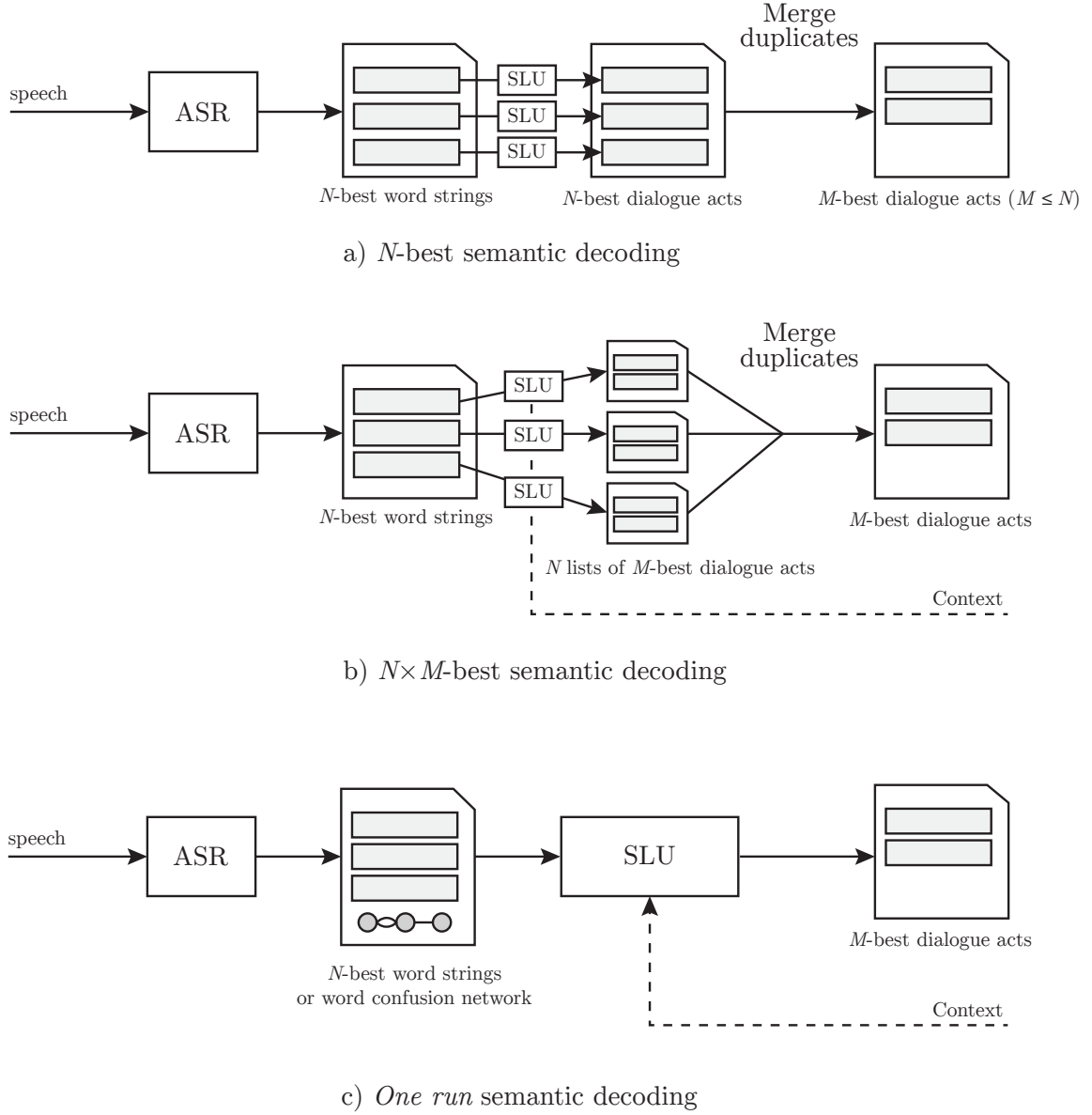


Figure 4.1: Semantic decoder configurations for generating an M -best list of dialogue act hypotheses:

- a) Each ASR hypothesis is decoded to give a single dialogue act. These are then weighted by the corresponding probabilities from the ASR, and duplicates are merged.
- b) Using a statistical semantic decoder, each ASR hypothesis generates an M -best list of dialogue acts. The semantic decoders extract features from the individual hypotheses, and optionally the dialogue context. The M -best lists are weighted by the ASR probabilities and then merged to give a single M -best list.
- c) Features are extracted from the whole ASR N -best list or word confusion network, and optionally the dialogue context. A statistical decoder uses these features and outputs an M -best list of dialogue acts.

reference label (dialogue act type or slot binding). To take account of the confidence scores of the hypotheses, each data point in the Support Vector Machine (SVM) training algorithm is assigned a misclassification cost proportional to the posterior probability of the hypothesis. A major limitation of this approach is that the size of the training set is multiplied by k . The training time of an SVM depends on the training set size R roughly as $O(R^3)$, so k must be kept small.

In one run semantic decoding, the SLU component is run once, rather than N times as in the previous configurations. Here the STC model is used. Rather than extracting features from individual ASR hypotheses, features are extracted from the entire N -best list or word confusion network. These features are then used to directly output an M -best list of the top dialogue acts in rank order according to equation (3.9). The following two sections describe *weighted sum* features to represent the N -best list, and *word confusion network* features to represent the confusion network for use in one run semantic decoding.

4.1.1 Weighted Sum Features

The n -gram features from each of the top N ASR hypotheses are weighted by their posterior probability and then summed, in an attempt to summarise the information in the N -best list. This representation can be written as:

$$f_i = \sum_{j=0}^{N-1} C_{hyp_j}(n\text{-gram}_i) \cdot p_j$$

where f_i is the i^{th} element of the feature representation, p_j is the posterior probability of the j^{th} ASR hypothesis in the N -best list hyp_j , and $C_{hyp_j}(n\text{-gram}_i)$ is the count of how many times $n\text{-gram}_i$ occurs in hypothesis hyp_j .

4.1.2 Word Confusion Network Features

Recall from section 2.1, the word confusion network is a representation of the full posterior distribution from the ASR. Word confusion networks allow for efficient calculation of the quantities $\mathbb{E}(C_u(n\text{-gram}_i))$, the expected frequency of $n\text{-gram}_i$ in the utterance (if the n -gram only appears in one path in the graph, then this is just its probability of occurrence) (Mangu et al., 2000).

The *CNet decoder* uses features derived from the word confusion network:

$$f_i = \mathbb{E}(C_u(n\text{-gram}_i))^{1/|n\text{-gram}_i|}$$

where $|n\text{-gram}_i|$ is the number of words in $n\text{-gram}_i$. The exponentiation is a normalisation included to compensate for the fact that longer word sequences are always less likely than their subsequences. Without this normalisation, weights for the longer n -grams would have to be higher to have equal influence in classification. The weight penalisation in the SVM quadratic program would then tend to avoid placing weight on potentially useful longer n -grams.

4.1.3 Dialogue Context Features

The STC framework can easily incorporate arbitrary features that may be relevant to decoding a spoken utterance. There may be some useful information in the context of the dialogue that could be exploited by these models. For example if the system has just requested a slot, then it is more likely for the user to inform this slot over others.

To investigate the effectiveness of using the dialogue context in improving the robustness of the semantic decoder, context features f_i^c are extracted from the last system act, D_m :

$$f_i^c = \begin{cases} 1 & \text{if } x_i \in D_m \\ 0 & \text{otherwise} \end{cases} \quad (4.1)$$

where x_0, x_1, \dots is an enumeration of all possible system dialogue act types and slot bindings. The final representation of a user utterance when using this context is then a concatenation of the original features and the context features. This allows the models to learn a dependence on the last act generated by the machine.

4.2 Off-line Evaluation

The in-car SLU corpus presented in section 3.2.1 is again used to evaluate different semantic decoder configurations in an off-line evaluation. Table 4.1 presents the results of this evaluation, giving the *F-score* and Item Cross Entropy (ICE) scores achieved on the test corpus.

The $N \times M$ -best and one run semantic decoding configurations in all cases give substantially improved ICE scores when compared to N -best semantic decoding. This suggests that these methods do indeed provide higher quality M -best lists. It is expected that $N \times M$ semantic decoding with the triangular CRF decoder could show a similar improvement. However, the implementation used (*TriCRF*) does not readily allow for outputting M -best lists of dialogue acts for single input word sequences.

Features	Trained on	Context	F-score	ICE	
<i>a) N-best semantic decoding</i>					
Phoenix	–	✗	0.694 ± 0.012	2.784 ± 0.116	(1)
Triangular CRF	top hypothesis	✗	0.674 ± 0.012	2.747 ± 0.112	(2)
STC	top hypothesis	✗	0.691 ± 0.012	2.561 ± 0.108	(3)
<i>b) $N \times M$-best semantic decoding</i>					
n -grams from N -best list hypotheses	top hypothesis	✗	0.692 ± 0.012	1.790 ± 0.065	(4)
	top 2 hypotheses	✗	0.703 ± 0.012	1.719 ± 0.068	(5)
	top hypothesis	✓	0.725 ± 0.011	1.529 ± 0.062	(6)
	top 2 hypotheses	✓	0.740 ± 0.011	1.499 ± 0.064	(7)
<i>c) One run semantic decoding</i>					
Weighted sum	Full N -best list ($N = 10$)	✗	0.708 ± 0.012	1.760 ± 0.074	(8)
		✓	0.742 ± 0.011	1.497 ± 0.066	(9)
Word confusion network	Full confusion network	✗	0.730 ± 0.011	1.680 ± 0.062	(10)
		✓	0.767 ± 0.011	1.431 ± 0.063	(11)

Table 4.1: Off-line evaluation of semantic decoders on ASR output. Results are written $\mu \pm 1.96\sigma$ where μ is the estimate of the mean over the utterances in the test corpus and σ the standard error. Results are split according to the three decoding configurations described in figure 4.1. N -best semantic decoding results are reproduced from table 3.3 for comparison. Row 8 is the CNet decoder.

Standard n -gram features are evaluated with k (the number of ASR hypotheses used for training) set at 1 and 2. Training complexity becomes an issue for higher values of k , and the increase in performance is negligible. The pay-off for increasing k was found to be higher for smaller training corpora. The systems trained on these features (rows 4 and 5) achieve F-scores comparable to Phoenix, and significantly smaller (i.e. better) ICE scores. The decrease in ICE score resulting from increasing k from 1 to 2 is quite substantial. Incorporating the last system act context features (rows 6 and 7) increases the F-scores to be higher than the Phoenix result.

The *one run* decoders trained on weighted sum features (rows 8 and 9) perform similarly to the $k = 2$ system (rows 5 and 7), suggesting that this is a reasonable method of summarising the information in the N -best list. Recall this representation avoids the problem of multiplying the size of the training set.

The decoders using word confusion network features (rows 10 and 11) perform well in comparison to the others. The F-score of the context independent decoder (row 10) is better than that of any other context independent decoder. The context dependent decoder (row 11, the CNet decoder) scored better than any other system for the F-score and ICE metrics. The word confusion network features are comparable to weighted sum features in the limit of increasing k , the number of top ASR hypotheses used. Intuitively, there is more information in these features, as they may pick out n -grams that do not appear in the top N hypotheses and furthermore assign all n -grams weights that more accurately reflect the estimate of the expected n -gram counts.

Figure 4.2 shows an example where the keyword ‘*west*’ is not in the top 10 ASR hypotheses, causing the hand-crafted grammar and CRF decoder to fail. The word ‘*west*’ is found (although with a low weight) in the confusion network, and the statistical models have learnt typical confusions of the speech recogniser, allowing it to give some weight to the correct hypothesis *inform(area=west)*. The last system act in the dialogue was asking the user to select between the west area and any area, so the model with context puts an even higher weight on the correct hypothesis.

To demonstrate that the CNet decoder (row 11) is more robust to noise and poor ASR performance than the Phoenix baseline, polynomial regressions were run for the two systems predicting ICE and F-score from the utterance Word Error Rate (WER) in the test set. Figure 4.3 plots these regressions. For the F-score, a degree 2 polynomial was found to model the data best, and for ICE a linear regression was found to be best using F-tests. The regressions suggest that the statistical decoder degrades significantly more gracefully when faced with speech recognition errors than the hand-crafted grammar.

Last system output: Sorry, do you want something in the west or you don't care?
Last system act: *select(area=west, area=Dontcare)*
User response: west side of town please
ASR N-best list: what kind of town please, what what kind of town please,
 kind of town please, etc.

M-best dialogue acts:

CNet decoder:	CNet decoder, without context features:
0.79 <i>inform(area=west)</i>	0.37 <i>null()</i>
0.15 <i>inform(area=north)</i>	0.32 <i>inform(area=west)</i>
0.05 <i>request(food, area=west)</i>	0.14 <i>inform(area=north)</i>
0.01 <i>null()</i>	0.09 <i>reqalts()</i>
	0.07 <i>request(food)</i>
	0.01 <i>inform(area=centre)</i>
Phoenix grammar:	Triangular CRF:
0.96 <i>null()</i>	0.87 <i>null()</i>
0.04 <i>request()</i>	0.13 <i>request()</i>

Figure 4.2: Illustrative example of semantic decoder output.

4.3 Live User Evaluation

A user trial was run to investigate the effect of using the best statistical decoder found in the previous section, the CNet decoder (table 4.1, row 11), in the context of an end-to-end dialogue system. The hand-crafted Phoenix grammar (Table 4.1, row 1) was used to provide a baseline.

4.3.1 Experimental Setup

One hundred native speakers of American English were recruited using *Amazon Mechanical Turk*TM. Each was asked to use the dialogue system to find a restaurant in Cambridge matching a set of constraints, and to then request some details. Some tasks involved specifying constraints that should be relaxed in case no matching restaurant was found, and tasks could also require the user to seek alternatives beyond the first restaurant offered by the system. An example of a typical task was: ‘Try to find a Chinese restaurant in the west, if there is none then try Thai food. Get the phone number and address.’ After a dialogue, the participant was asked whether or not they got the information they needed, and if they agreed then the dialogue was recorded as being successful.

Participants were randomly allocated either a system using the Phoenix grammar, or one

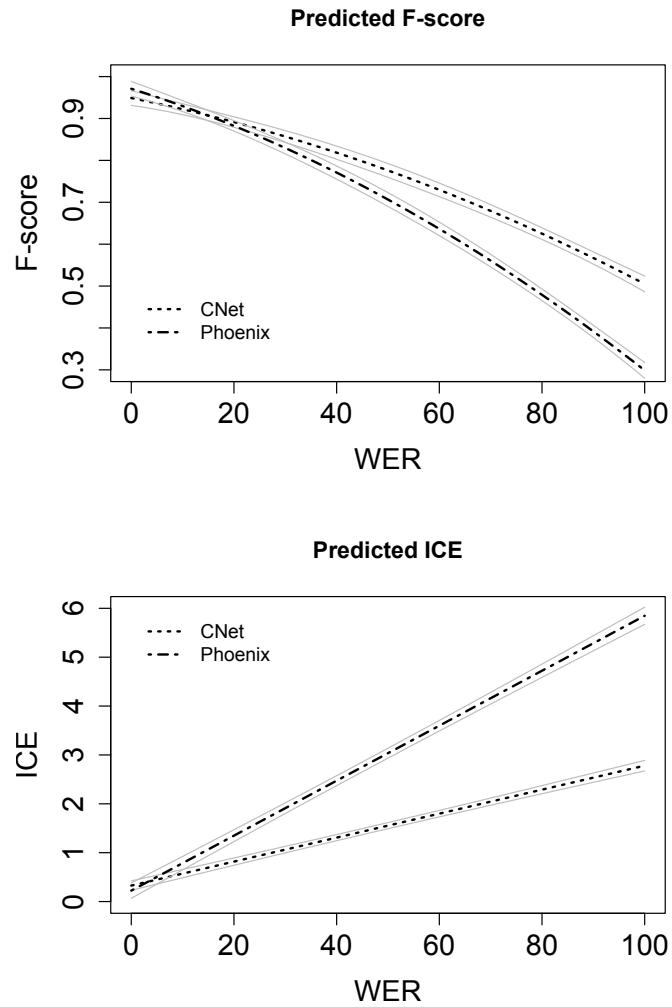


Figure 4.3: Regressions of F-score and ICE against WER. Grey lines show margins of 2 standard errors. CNet decoder is shown to degrade significantly more gracefully as the noise as measured by the WER increases.

using the CNet decoder. Both dialogue systems use the Bayesian Update of Dialogue State (BUDS) framework to track the dialogue belief state, treating the dialogue planning process as a Partially Observable Markov Decision Process (POMDP) (Thomson and Young, 2010). The dialogue policy for each system was optimised to maximise the dialogue reward R ,

$$R = 20 \cdot \text{success} - \text{Number of user turns} \quad (4.2)$$

where success = 1 if the dialogue is successful according to the participant, and 0 otherwise. This reward function provides a measure of dialogue quality reflecting the design objective of achieving a high success rate and short dialogues. Each policy was trained using the Natural Actor Critic learning algorithm with a simulated user and a built-in error model. Each error model was separately trained to reflect the type of confusions each decoder makes, using real example confusions and the technique described in Thomson et al. (2012).

Note that unlike the off-line corpus, which used noisy in-car dialogues, the live trial was conducted using relatively clean telephone calls and a different set of acoustic models optimised for this domain. The WER over the 924 trial dialogues was 20.1%, while the WER in the off-line corpus was 37.0%.

4.3.2 Results

The results of the trial are shown in table 4.2. The F-scores, ICE scores and Dialogue Reward achieved by the CNet decoder are significantly better than the Phoenix grammar. The raw success rate is also higher for the CNet decoder although the difference is not statistically significant. The average dialogue length is shorter by half a *turn* on average, and this is significant.

Figure 4.4 presents a logistic regression of success rate against the WER for the two evaluated dialogues in the user trial, as well as linear regressions of the dialogue reward and number of turns against the WER for successful dialogues. There is no significant difference in the correlation of success rate and WER. However, the CNet decoder dialogue system is found to degrade in performance more gracefully when faced with poor speech recognition results as measured by the dialogue reward and number of turns in successful dialogues.

Overall the differences between the two decoders are not as pronounced as in the off-line evaluation because the dialogues in the trial were at much lower word error rates. The high success rates achieved in the user trial are indicative of this. Given the evidence of the off-line evaluation, it is hypothesised that the advantages of the CNet decoder would become more pronounced in more challenging scenarios such as in a car using an open far-field microphone.

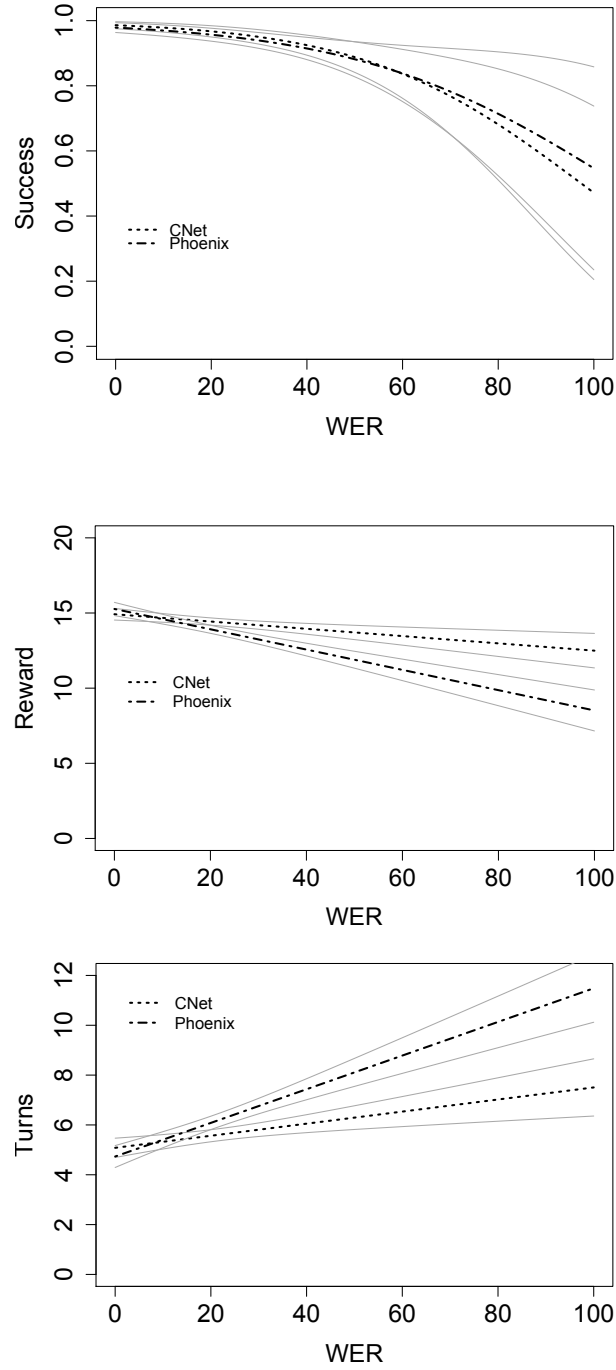


Figure 4.4: Logistic regression of success rate against WER, and linear regressions of dialogue reward and number of turns against WER for CNet and Phoenix decoders in successful dialogs. Regressions are done on successful dialogues as both systems performed at almost identical success rates, and the turns and dialogue reward are distributed bi-modally, split by dialogue success. Grey lines show margins of 2 standard errors.

	Phoenix	CNet
Dialogues	456	468
F-score	0.795 ± 0.01	0.822 ± 0.01
ICE	2.016 ± 0.114	1.264 ± 0.077
Success Rate (%)	94.3 ± 1.0	94.7 ± 1.1
Dialogue length (turns)	6.25 ± 0.15	5.79 ± 0.13
Dialogue Reward	12.61 ± 0.28	13.14 ± 0.28

Table 4.2: Results of user trial. Errors are the standard error in the mean. Note that the turns are 2.0 less than reported in Henderson et al. (2012), as here the two final turns of the dialogue are not counted, where the system provides a code for the *Amazon Mechanical Turk*TM feedback form.

4.4 Conclusions

While the semantic decoders evaluated in chapter 3 all performed roughly equally on clean text, this chapter has found benefits from exploiting more information from the ASR output when the decoders are used in a spoken dialogue system. Although N -best semantic decoding provides a simple way to use conventional decoders in a statistical dialogue system, significant gains are found by using $N \times M$ -best and one pass semantic decoding configurations.

Building on the STC framework proposed by Mairesse et al., this chapter has described a statistical Confusion Network (CNet) semantic decoder, which is applied directly in a single run to features extracted from a word confusion network. It has been shown through off-line evaluation on a dialogue corpus collected in noisy conditions that the CNet decoder approach outperforms both a hand-crafted Phoenix-based decoder and an STC decoder operating in N -best mode. Furthermore it does this for both a 1-best F-measure metric and a full distribution cross-entropy metric, demonstrating that the approach improves both the quality of individual semantic interpretations and the full distribution over all interpretations. It has also been shown that the addition of context is both simple to implement in this framework and effective in further improving performance. Finally, it has been shown via a user trial that the performance advantages indicated by off-line evaluation do translate into improved overall performance when used in a live dialogue system.

CHAPTER 5

DIALOGUE STATE TRACKING

In a statistical dialogue system, Dialogue State Tracking (DST) promises to provide robustness to speech recognition and Spoken Language Understanding (SLU) errors by providing a well calibrated distribution over possible dialogue states at each *turn* in the dialogue (see section 2.3). SLU gives a local classification of a spoken utterance at each turn, specifying e.g. a distribution over slot-value constraints being given by the speaker. On the other hand, a dialogue state tracker must use the history of the dialogue up to the latest turn to keep track of the dialogue state. This includes remembering constraints given earlier in the dialogue, dealing with confirmations from the system (“*Did you want Indian food?*”), and understanding how different dialogue acts affect the state.

After formulating the task, this chapter presents a selection of methods for DST. Note that a full review of methods for DST was given in section 2.3. The main contribution of this chapter is a *discriminative* model for DST called the *word-based* Recurrent Neural Network (RNN) tracker, which uses an RNN to model dialogue sequences. The tracker maps directly from the posterior distribution over words provided by the Automatic Speech Recognition (ASR) to a distribution over dialogue states. Word-based tracking is the first method for DST that avoids the need for an intermediate semantic representation, the need to develop a separate SLU component, and any inherent bottleneck from the SLU step.

Chapter 6 will then present an evaluation of these methods both off-line in a corpus-based setting and online in a live user trial.

5.1 Formulation of Dialogue State Tracking

This section defines the problem of DST as it is presented in the Dialog State Tracking Challenge (DSTC) 2 and 3 evaluations (Williams et al., 2014). The DSTCs are blind evaluations of DST, using common data and evaluation suites, described further in section 6.1. The following formulation of the DST task is adopted for the remainder of this thesis.

In this thesis, DST is evaluated using dialogues in the restaurant and tourist information domains (see appendix A). Users search for venues by specifying constraints, and may ask for information such as the phone number of already offered venues. The dialogue state is formulated in a manner that is general to information browsing tasks such as this.

The dialogue state at each turn consists of three components:

- The *goal constraint* for each informable slot $s \in S_{inf}$ (see appendix A for notation). This is either an assignment of a value $v \in V_s$ that the user has specified as a constraint, or is a special value — either *Dontcare*, which means the user has no preference, or *None*, which means the user is yet to specify a valid goal for this slot.
- A set of *requested slots*, i.e. those slots whose values have been requested by the user, and should be informed by the system. This is a subset of S_{req} .
- The current dialogue *search method*. This is one of
 - *by constraints*, if the user is attempting to issue a constraint,
 - *by alternatives*, if the user is requesting alternative suitable venues,
 - *by name*, if the user is attempting to ask about a specific venue by its name,
 - *finished*, if the user wants to end the call,
 - or *none* otherwise.

These dialogue state components combine to give everything that a dialogue manager should require in order to select its next action. Indeed the marginal distributions over these state components are used as the input for the dialogue *policy* in the Bayesian Update of Dialogue State (BUDS) system (see section 2.4). Note that the set of possible dialogue states is fully specified by the domain (the sets of slots and their possible values).

Figure 5.1 gives an illustrative example dialogue with the dialogue state labelled at each turn. In the example dialogue, turn 2 demonstrates an addition to the goal constraints. In turn 3, the goal constraint for the *area* slot changes, and the search method changes from *by constraints* to *by alternatives*. The last turn demonstrates a non-empty set of requested slots.

Actual System and User turn	SLU output	Dialogue State Labels
Turn 1		
S: What part of town did you have mind? <i>request(area)</i>	0.7 <i>null()</i> 0.2 <i>inform(food='north african')</i> 0.1 <i>inform(area=north)</i>	goal constraints <i>area=north</i> requested slots – search method <i>by constraints</i>
U: The North uh area <i>inform(area=north)</i>		
Turn 2		
S: Which part of town? <i>request(area)</i>	0.8 <i>inform(area=north, pricerange=cheap)</i> 0.2 <i>inform(area=north)</i>	goal constraints <i>area=north</i> <i>pricerange=cheap</i> requested slots – search method <i>by constraints</i>
U: A cheap place in the North <i>inform(area=north, pricerange=cheap)</i>		
Turn 3		
S: Da Vinci Pizzeria is a cheap place in the North of town. <i>inform(name='Da Vinci Pizzeria', area=north, pricerange=cheap)</i>	0.7 <i>reqalts(area=west)</i> 0.3 <i>reqalts()</i>	goal constraints <i>area=west</i> <i>pricerange=cheap</i> requested slots – search method <i>by alternatives</i>
U: Do you have anything else, but in the West? <i>reqalts(area=west)</i>		
Turn 4		
S: Cocum is a cheap place in the West. <i>inform(name=cocum, area=west, pricerange=cheap)</i>	0.6 <i>request(phone, address)</i> 0.3 <i>request(phone)</i> 0.1 <i>request(address)</i>	goal constraints <i>area=west</i> <i>pricerange=cheap</i> requested slots <i>address</i> <i>phone</i> search method <i>by alternatives</i>
U: What is their phone number and address? <i>request(address, phone)</i>		

Figure 5.1: Example dialogue with dialogue state labels. The left column shows the actual system output and user input. The second column shows example SLU M -best hypotheses and their scores. In practice, up to 10 SLU M -best hypotheses are output.

A tracker must use information up to a given turn in the dialogue, and output a probability distribution over dialogue states for the turn. Trackers output separately the distributions for goal constraints, requested slots and the search method. The distribution over a component of the dialogue state is sometimes referred to as the tracker’s *belief*.

5.2 Baseline Methods for Tracking

Three baseline dialogue state trackers are presented. These simple rule-based approaches provide a comparison to the trackers described in the following sections. Many approaches for DST proposed in the literature are by contrast statistical models learnt from data (including the approach presented in this thesis using RNNs). The baseline trackers proved strong competition in the DSTCs, with only around half of the entries in DSTC 3 beating the top baseline approach in terms of joint goal constraint *accuracy* (Henderson et al., 2014d).

Source code for all the baseline systems has been made public on the DSTC website¹.

5.2.1 SLU Evidence Observations

The baseline trackers take as input the distribution of dialogue acts given by the SLU at each turn. This distribution is first converted into a set of observations o_i for each component of the dialogue state i :

- $o_{g_s}(v)$ for each informable slot $s \in S_{\text{inf}}$ and $v \in V_s$ – the observation of the user providing the goal constraint $s = v$.
- o_{r_s} for each requestable slot $s \in S_{\text{req}}$ – the observation of the user requesting the slot.
- $o_m(\text{method}_i)$ – the observation of the user specifying the search method as method_i .

Following the notation of chapter 3, suppose the output of the SLU is the distribution over dialogue act hypotheses D_{hyp_i} with corresponding probabilities p_i for $i = 0, \dots, n-1$, where D_{hyp_i} is given by the dialogue act type $\text{d-type}_{\text{hyp}_i}$ and the set of slot bindings X_{hyp_i} . The observations are then calculated as follows:

$$o_{g_s}(v) = \sum_{i=0}^{n-1} \begin{cases} p_i & \text{if } s = v \in X_{\text{hyp}_i} \\ & \text{or } \text{d-type}_{\text{hyp}_i} = \textit{affirm} \text{ and last system act confirmed } s = v \\ 0 & \text{otherwise} \end{cases} \quad (5.1)$$

¹<http://camdial.org/~mh521/dstc/>

Last system act, a <i>confirm(area=east)</i>	
SLU M-best list	
0.6	<i>inform(food=indian)</i>
0.3	<i>request(phone, food=italian)</i>
0.2	<i>affirm()</i>
0.1	<i>reqalts()</i>
$o_{g_{food}}(\textit{indian})$	= 0.6
$o_{g_{food}}(\textit{italian})$	= 0.3
$o_{g_{area}}(\textit{east})$	= 0.2
$o_m(\textit{by constraints})$	= 0.9
$o_m(\textit{by alternatives})$	= 0.1
$o_{r_{phone}}$	= 0.3

Figure 5.2: Example demonstrating the calculation of the SLU observation quantities, o_i . The observations not listed above are all 0. The top two acts in the SLU output contribute evidence for the *indian* and *italian* hypotheses for the *food* goal constraint. In the context of the last system act, the *affirm()* hypothesis from the SLU provides evidence for the *area=east* goal constraint hypothesis. All the hypotheses imply a search method of *by constraints* apart from the *reqalts()* act, which implies *by alternatives*.

for each value $v \in V_s$.

$$o_{r_s} = \sum_{i=0}^{n-1} \begin{cases} p_i & \text{if d-type}_{\text{hyp}_i} = \textit{request} \text{ and } s \in X_{\text{hyp}_i} \\ 0 & \text{otherwise} \end{cases} \quad (5.2)$$

$$o_m(\textit{method}_j) = \sum_{i=0}^{n-1} \begin{cases} p_i & \text{if d-type}_{\text{hyp}_i} \text{ implies search method}_j \\ 0 & \text{otherwise} \end{cases} \quad (5.3)$$

for each search method, \textit{method}_j .

A worked example is given in figure 5.2 to illustrate the calculations. The o_i quantities are identical to the p^+ quantities calculated in Sun et al. (2014b).

5.2.2 One-best Baseline

The *one-best baseline* gives a single hypothesis for each component of the dialogue state, whose value is the top scoring suggestion so far in the dialogue.

For slot $s \in S_{\text{inf}}$, it reports the value for the goal constraint at turn t to be:

$$\arg \max_v \max_{t' \leq t} o_{g_s, t'}(v) \quad (5.4)$$

with score the value of the max. The remaining probability mass is assigned to the *None* hypothesis. Here the observation o_i at turn t is written $o_{i, t}$. Similarly the one-best baseline reports the search method as

$$\arg \max_{\text{method}} \max_{t' \leq t} o_{m_s, t'}(\text{method}) \quad (5.5)$$

again with score the value of the max. Lastly for requested slots, the tracker assigns a score of:

$$\max_{\text{lastInf}(s, t) \leq t' \leq t} o_{r_s, t'} \quad (5.6)$$

to each requestable slot $s \in S_{\text{req}}$, where $\text{lastInf}(s, t)$ is the last time the system informed the slot s , previous to turn t .

The one-best tracker implements a traditional non-statistical approach to DST, which does not make use of the full distribution given by the SLU. Note that this tracker does not account well for goal constraint changes; the hypothesised value for a slot will only change if a new value occurs with a higher confidence. It also does not accumulate evidence from the SLU over time.

5.2.3 Focus Baseline

The *focus baseline* includes a simple model of evidence accumulation and changing goal constraints (the user changing their focus of attention). This tracker outputs a distribution over dialogue states at each turn.

The probability over goal constraints at turn t , $P(g_s = v)_t$ is initialised with $P(g_s = v)_{-1} = 0 \ \forall v \in V_s$ (so the probability of *None* is initialised to 1). It is then updated as follows:

$$P(g_s = v)_t = P(g_s = v)_{t-1} \left(1 - \sum_{v' \in V_s} o_{g_s, t}(v') \right) + o_{g_s, t}(v) \quad (5.7)$$

This interpolates between not changing the distribution in the case that there is no information about the goal constraint in turn t , and copying the SLU observations o_{g_s} if the SLU is certain the constraint has been given.

The search method is tracked in the same manner. The initialisation is $P(m = \text{method}_i)_{-1} = 0$ for every search method except $P(m = \text{none})_{-1} = 1$. The update equation is:

$$P(m = \text{method}_i)_t = P(m = \text{method}_i)_{t-1} \left(1 - \sum_j o_{m,t}(\text{method}_j) \right) + o_{m,t}(\text{method}_i) \quad (5.8)$$

Lastly for requested slots, the focus tracker outputs a probability score $P(r_s)_t$ for each $s \in S_{\text{req}}$ at turn t . This is initialised so $P(r_s)_{-1} = 0$, and updated with:

$$P(r_s)_t = o_{r_s,t} + P(r_s)_{t-1}(1 - o_{r_s,t}) \left(\begin{cases} 0 & \text{if machine action at turn } t \text{ informs slot } s \\ 1 & \text{otherwise} \end{cases} \right) \quad (5.9)$$

This implements the same type of interpolation between previous output and new SLU observations, with the added behaviour that the output is reset to 0 if the machine has informed the slot.

5.2.4 HWU Baseline

Another baseline tracker, based on the rule-based tracker presented in Wang and Lemon (2013), is also included in the evaluation of the DSTCs. This tracker uses a selection of domain independent rules to update its outputs, similar to the focus baseline. One rule uses a simple learnt parameter called the noise adjustment, to adjust the SLU scores.

5.3 Generative Tracking with Dynamic Bayesian Networks

This section summarises the approach taken for DST in the BUDS system (Thomson and Young, 2010; Thomson, 2009). This has been the state of the art used for research in statistical Spoken Dialogue Systems (SDSs) in the Cambridge dialogue group for the past 4 years. BUDS defines a *generative Bayesian network*, which models how the hidden dialogue state produces noisy observations from the SLU, and changes from turn to turn. The notation presented here is slightly adapted from Thomson and Young (2010) and Thomson (2009) to better match the formulation of DST used in the DSTCs. A detailed introduction to Bayesian networks is presented in chapter 6 of Bishop (2006).

The Bayesian network used in BUDS models the evolution of the goal constraints $g_{s,t}$, the requested slots $r_{s,t}$ and the search method m_t . For each of these hidden random variables

there is a corresponding *true user act* variable, which represents the underlying action that the user took for that component of the dialogue state ($u_{g_s,t}$, $u_{r_s,t}$ and $u_{m,t}$). Figure 5.3 shows the structure of the network. The structure of the distributions has been slightly simplified in the following text, but full details of the exact structure used can be found in Thomson and Young (2010); Thomson (2009). The framework also allows for adding dependencies between the g_s variables in the network.

The network is a *dynamic Bayesian network*, meaning that the structure is identical inside individual time steps, there are no connections spanning more than one time step, and the structure connecting adjacent time steps is constant.

The observed variables in the network, the system actions a and the SLU observations o_i (see section 5.2.1), are shaded grey. The goal constraints, requested slots and search method random variables are connected to their values in the previous time step, as well as the system action. The corresponding conditional probability for the goal constraints is of the form:

$$P(g'_s = v' | g_s = v, a) = \begin{cases} 1 - \theta_{class(a,s)} & \text{if } v = v' \\ \theta_{class(a,s)} & \text{if } v \neq v' \end{cases} \quad (5.10)$$

where $0 < \theta_{class(a,s)} < 1$ is the probability of the user goal changing dependent on $class(a, s)$, the *class* of the machine action with respect to slot s . The classes used are: a is informing slot s , a is requesting slot s , a is a *hello* act, and other. This allows for learning that the user is more likely to change their goal in certain contexts. Similar conditional probabilities are used for the requested slots and search method.

The true user act variables are connected to the variables for their respective state components, with corresponding conditional probability:

$$P(u_i = v | g_i = v') = \begin{cases} 1 - \theta_i & \text{if } v = v' \\ \theta_i & \text{if } v \neq v' \end{cases} \quad (5.11)$$

where $0 < \theta_i < 1$ is the probability of the user not disclosing their true goal and is typically set to a small value.

Lastly the observations o_i are connected to the u_i variables. This is a deterministic connection forcing the marginal distribution of the u_i to be set to the observations from the SLU.

The θ parameters are parameterised by Beta distributions. The hyperparameters of these beta distributions, as well as other hyperparameters that parameterise the priors for the first time slice are hand-crafted to sensible values by the system designer. The hyperparameters

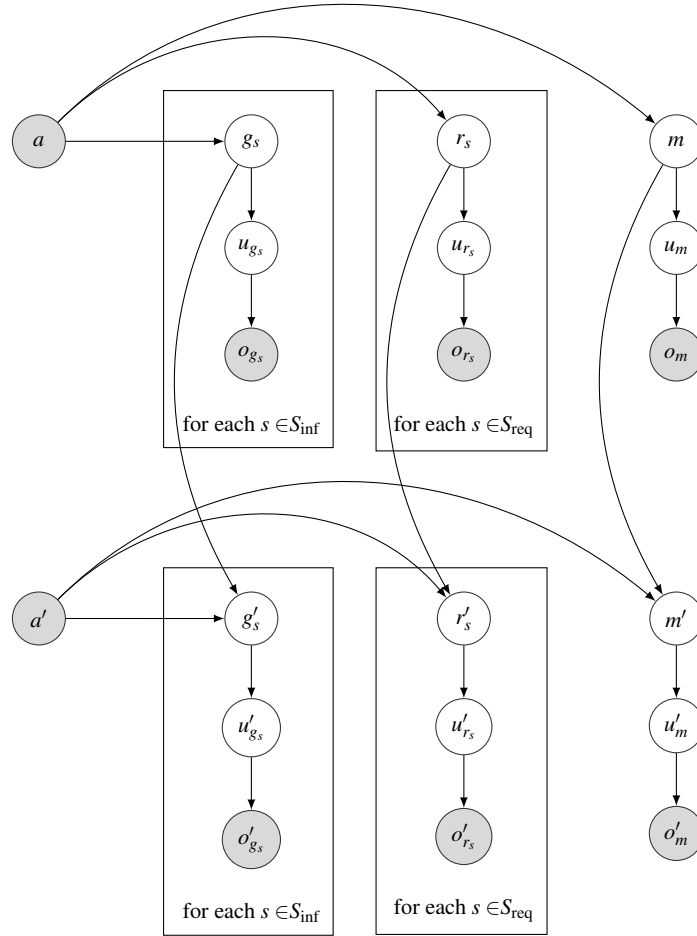


Figure 5.3: Dynamic Bayesian network structure for DST, showing two consecutive time steps. For brevity the t subscript is omitted and variables in the $t + 1$ time step are marked with a prime ($'$). Observed nodes are shaded grey. The connections in the graph encode the conditional independency relationships in the joint distribution of the random variables. The conditional probabilities are given in the text.

can also be learnt using expectation propagation and unlabelled dialogue corpora. Expectation propagation is used to infer marginal distributions over the g_s , r_s and m variables (the required output of the dialogue state tracker). This is an approximate Bayesian inference technique, which can be optimised to run quickly and accurately for these networks. More details on expectation propagation for parameter learning and inference is given in Thomson et al. (2010b).

5.4 Discriminative Dialogue State Tracking

There are several deficiencies of generative models, some of which were identified in Williams (2012). Generative approaches must model all the correlations in the input features, so they cannot easily exploit arbitrary but potentially useful features of the dialogue history. Any such features must be included into the user model requiring new structures and dependencies to be learnt or specified.

The generative models used for DST make many independence assumptions in order to make the models tractable, in terms of both computational complexity and the number of parameters that must be learnt. In particular, the current implementations do not model ASR and SLU error correlations, instead assuming independence for simplicity.

The assumption made by the generative models used for DST that the ASR and SLU errors are uncorrelated is unrealistic, and can lead to poor performance. Any given speaker will cause certain misrecognition patterns in the ASR that cannot currently be exploited. For example, a key concept may be consistently misinterpreted by the SLU. A generative model might accumulate more and more evidence for the wrong interpretation in a dialogue as the user repeats themselves and the same misinterpretation is made. Ideally the system would learn to not trust certain concepts so readily, and to identify when a misinterpretation is likely to have been made. For example, if a certain word is *always* misinterpreted as another, then existing generative models will consistently make the same mistake. However, an ideal dialogue state tracker would learn this confusion and track the dialogue state accordingly. It would be possible to address this in generative models by e.g. increasing the complexity of the generative model's distribution of observations given the true user action $P(o | u)$. Note that this may cause issues of data sparsity when learning the new parameters.

Discriminative models address these issues by directly modelling the distribution over the dialogue state given arbitrary and possibly correlated input features. A remaining advantage of generative approaches is that their internal processes remain relatively easy to understand and inspect. The user model generally defines a *generative story* of how an internal hidden dialogue state *generates* observations. Some discriminative models can be

somewhat opaque, particularly the huge networks of interconnected neurons used in RNNs and neural networks. There are however techniques for inspecting a learnt RNN, such as looking at gradients of key neurons with respect to particular input patterns. Other discriminative approaches are easier to inspect, for example human-understandable decision trees are the key building block of the discriminative ranker used in Williams (2014).

This section will describe some discriminative methods for DST. The following section will then present how RNNs can be applied to discriminative DST, which is the main contribution of this chapter.

5.4.1 Using Static Classifiers

DST can be considered as specifying the distribution:

$$P(s_t | o_0, \dots, o_t) \quad (5.12)$$

where s_t is the dialogue state at the current turn t , and o_0, \dots, o_t is the sequence of observations from the SLU and machine actions up to and including the current turn. One key issue with this approach is that it is necessary to extract a fixed dimensional feature vector to summarise a sequence of arbitrary length observations o_0, \dots, o_t .

Given a feature representation, this distribution has been modelled using a variety of discriminative models including maximum entropy linear classifiers (Bohus and Rudnicky, 2006; Lee and Eskenazi, 2013; Metallinou et al., 2013; Williams, 2012, 2013), neural networks (Henderson et al., 2013; Ren et al., 2014b,a; Sun et al., 2014b) and ranking models (Williams, 2014).

This section presents in some detail the method used in Metallinou et al. (2013), which is fairly representative of this class of techniques. Consider tracking the correct value for the user’s goal constraint for a given slot s . Metallinou et al. extract a feature vector \mathbf{f} from the observation and a set of vectors \mathbf{f}_v for each v that has appeared in the SLU so far. The \mathbf{f}_v features are intended to convey information about the correctness of the v hypothesis for the goal constraint on slot s . The \mathbf{f} features are of general interest, and are also used to calculate the probability of the *None* hypothesis.

The general \mathbf{f} provide aggregate information about the dialogue history, including:

- the size of the SLU M -best list in the latest turn
- the entropy of the SLU M -best list in the latest turn
- the posterior scores from the ASR in the latest turn

- the mean and variance of the SLU M -best list lengths in the dialogue history
- the number of distinct SLU results obtained so far and the entropy of the corresponding probabilities

For a value v , the value specific features \mathbf{f}_v include:

- information about the latest turn:
 - the rank and probability score of the $s = v$ hypothesis in the SLU list
 - the difference between the probability score of the $s = v$ hypothesis and the top scoring hypothesis in the SLU list
- information from the dialogue history:
 - the number of times the hypothesis $s = v$ has appeared in the SLU so far
 - the number of times the hypothesis $s = v$ has appeared in the SLU so far at each particular rank
 - the sum and average of the confidence scores of SLU results containing $s = v$
 - the number of possible past user negations or confirmations of $s = v$
- information about likely ASR errors and confusability:
 - estimates for the likelihood of $s = v$ being correctly identified by the SLU, estimated on held-out training data
 - a similar estimate for the prior probability of $s = v$ appearing in an SLU M -best list, and at specific rank positions in the list
 - similar estimates for how likely $s = v$ is to be confused with other hypotheses $s = v'$

The approach of looking at sums and averages of features to summarise the dialogue history in a fixed length vector is typical of these methods. In Henderson et al. (2013) an alternative approach is used, involving a sliding window that runs over features $\mathbf{f}_{v,t}$, which depend on the value v and turn t .

Though the features \mathbf{f}_v are of a fixed dimension, there is a dynamic number of such features calculated at each turn as more values v appear in the SLU hypotheses. This is dealt with in the static classifier by tying the parameters that interact with the \mathbf{f}_v vectors.

5.4.2 Using Conditional Random Fields

The use of static classifiers essentially ignores the sequential nature of the input, using feature functions that attempt to summarise the sequence. A Conditional Random Field (CRF) can be used to do sequential labelling of the dialogue (Kim and Banchs, 2014; Lee, 2013). A linear-chain CRF is used (very similar to the CRF method described in section 3.1.2 for SLU), which learns the conditional distribution:

$$P(s_0, \dots, s_t \mid o_0, \dots, o_t) \quad (5.13)$$

This is then marginalised to give the distribution for the latest state s_t . Features like those used in section 5.4.1 to extract information about hypotheses for the current turn are used, except these are calculated for all turns in the history.

One key issue with using CRFs for DST is that continuous features must be quantised, as the CRF modelling techniques in all published work on DST so far require discrete features.

5.5 Discriminative Tracking with Recurrent Neural Networks

RNNs are discriminative models that can deal with the sequential nature of the dialogue without requiring features to summarise the history. Unlike the proposed CRF models, they can easily deal with continuous features.

This section describes a Recurrent Neural Network (RNN) model for DST, a discriminative approach to state tracking first presented in Henderson et al. (2014a). The RNN takes as input at each turn a feature representation of the last machine action and the SLU output from the user. The RNN updates an internal *memory* and outputs a new distribution over the dialogue states. The following section provides a definition of this model.

5.5.1 Feature Representation

Extracting n -gram type features from dialogue acts provides the feature representations needed for input into the RNN. The process of converting a dialogue act to a list of n -grams is outlined in table 5.1.

This provides a method of representing the dialogue acts output by the system. The SLU M -best dialogue act list is encoded in the same way except that the n -grams from each hypothesis are weighted by the corresponding probabilities, and summed to give a single feature vector.

A combined feature representation of both the SLU M -best list and the last machine act

SLU M -best list		Machine act	
<i>inform(area=east)</i>	0.9	<i>confirm(area=east)</i>	
<i>inform(area=west)</i>	0.1		
<i>inform area east</i>	0.9	<i>confirm area east</i>	1.0
<i>inform area west</i>	0.1	<i>confirm area</i>	1.0
<i>inform area</i>	1.0	<i>area east</i>	1.0
<i>area east</i>	0.9	<i>confirm</i>	1.0
f <i>area west</i>	0.1	<i>area</i>	1.0
<i>inform</i>	1.0	<i>east</i>	1.0
<i>area</i>	1.0		
<i>east</i>	0.9		
<i>west</i>	0.1		
<i>inform <slot> <value></i>	1.0	<i>confirm <slot> <value></i>	1.0
<i><slot> <value></i>	1.0	<i>confirm <slot> <value></i>	1.0
<i><slot></i>	1.0	<i><slot> <value></i>	1.0
<i><value></i>	1.0	<i><slot></i>	1.0
f_s <i>inform area <value></i>	1.0	<i><value></i>	1.0
<i>area <value></i>	1.0	<i>confirm area <value></i>	1.0
<i><slot> east</i>	0.9	<i>area <value></i>	1.0
<i><slot> west</i>	0.1	<i>confirm <slot> east</i>	1.0
<i>inform <slot> east</i>	0.9	<i><slot> east</i>	1.0
<i>inform <slot> west</i>	0.1		
<i>inform <slot> <value></i>	0.9	<i>confirm <slot> <value></i>	1.0
<i><slot> <value></i>	0.9	<i><slot> <value></i>	1.0
f_v <i><value></i>	0.9	<i><value></i>	1.0
<i>inform area <value></i>	0.9	<i>confirm area <value></i>	1.0
<i>area <value></i>	0.9	<i>area <value></i>	1.0

Table 5.1: Extracting n -gram features from the dialogue acts in the SLU M -best list and the machine act, showing \mathbf{f} , $\mathbf{f}_{s=area}$ and $\mathbf{f}_{v=east}$. For all $v \notin \{east, west\}$ $\mathbf{f}_v = \mathbf{0}$.

is obtained by concatenating the vectors. This means that in table 5.1 the *area* feature from the SLU and the *area* feature from the machine act contribute to separate components of the final vector representation \mathbf{f} .

5.5.2 Generalisation to Unseen Dialogue States

One key issue in applying machine learning to the task of dialogue state tracking is being able to deal with states that have not been seen in training. For example, the system should be able to recognise any obscure food type that appears in the set of possible food types (V_{food}). A naïve neural network structure mapping dialogue act n -gram features to an updated distribution for the food slot, with no tying of weights, would require separate examples of each of the food types to learn what dialogue acts are associated with each. In reality however, dialogue act n -grams such as ‘*inform* $\langle value \rangle$ *food*’ and ‘ $\langle value \rangle$ *food*’ are likely to correspond to the hypothesis *food*=‘ $\langle value \rangle$ ’ for any food-type replacing ‘ $\langle value \rangle$ ’.

The approach taken here is to embed a network that learns a generic model of the updated belief of a slot-value assignment as a function of *tagged* (or *delexicalised*) features, i.e. features that ignore the specific identity of a value. This can be considered as replacing all occurrences of a particular value with a tag like ‘ $\langle value \rangle$ ’. Table 5.1 shows the process of creating the feature vectors, \mathbf{f}_s and \mathbf{f}_v from the untagged vector \mathbf{f} .

5.5.3 Recurrent Neural Network Structure

This section describes an RNN structure for tracking the goal constraint for a given informable slot $s \in S_{inf}$ throughout the sequence of a dialogue. The RNN can be thought of as a combination of Elman and Jordan types (see appendix D); at each turn it takes the previous output as input, but it also uses a hidden layer as a recurrent internal memory.

In what follows, the notation $\mathbf{a} \oplus \mathbf{b}$ is used to denote the concatenation of two vectors, \mathbf{a} and \mathbf{b} . The i^{th} component of the vector \mathbf{a} is written a_i .

The RNN takes the most recent dialogue turn (user input plus last machine dialogue act) as input, updating its internal memory and calculating an updated distribution over the values for the slot.

The RNN holds an internal memory, $\mathbf{m} \in \mathbb{R}^{N_{\text{mem}}}$, which is updated at each step. If there are N possible values for slot s (i.e. $N = |V_s|$), then the probability distribution output \mathbf{p} is in \mathbb{R}^{N+1} , with the last component p_N giving the probability of the *None* hypothesis. Recall the *None* hypothesis is that no value has been mentioned yet by the user for this slot’s goal constraint.

Figure 5.4 provides an overview of how \mathbf{p} and \mathbf{m} are updated in one turn to give the new

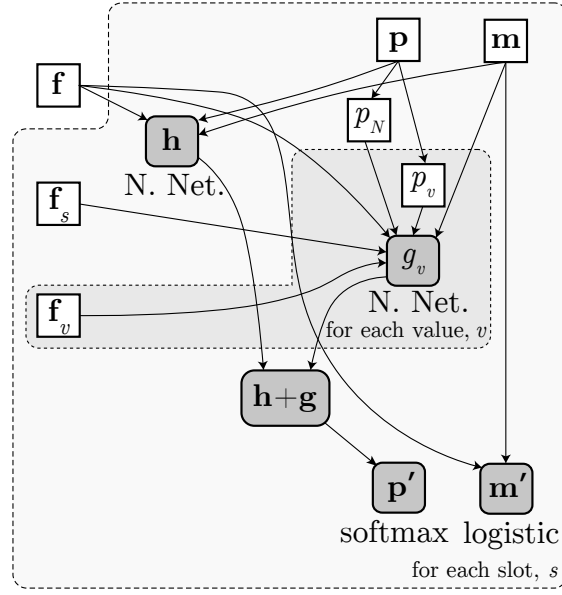


Figure 5.4: Calculation of \mathbf{p}' and \mathbf{m}' for one turn. This is an illustration of equations (5.14-5.17).

belief and memory, \mathbf{p}' and \mathbf{m}' . An alternative summary of the RNN structure is presented in figure 5.5, which uses a more conventional layout. In both diagrams, the slot s runs over all slots in S_{inf} and the value v over all values in V_s .

One part of the neural network is used to learn a mapping from the untagged inputs, full memory and previous beliefs to a vector $\mathbf{h} \in \mathbb{R}^N$, which goes directly into the calculation of \mathbf{p}' :

$$\mathbf{h} = \text{NNet}(\mathbf{f} \oplus \mathbf{p} \oplus \mathbf{m}) \in \mathbb{R}^N \quad (5.14)$$

where $\text{NNet}(\cdot)$ denotes a neural network function of the input. In this chapter all such networks have one hidden layer with a sigmoidal activation function. Appendix D explains this notation further.

The sub-network for \mathbf{h} requires examples of every slot value in training, and is prone to poor generalisation as explained in section 5.5.2. By including a second sub-network for \mathbf{g} that takes the feature vectors \mathbf{f}_s and \mathbf{f}_v as input, it is possible to learn behaviours that generalise across values. For each value v , a scalar component of \mathbf{g} is calculated using the neural network:

$$g_v = \text{NNet}(\mathbf{f} \oplus \mathbf{f}_s \oplus \mathbf{f}_v \oplus \{p_v, p_N\} \oplus \mathbf{m}) \in \mathbb{R} \quad (5.15)$$

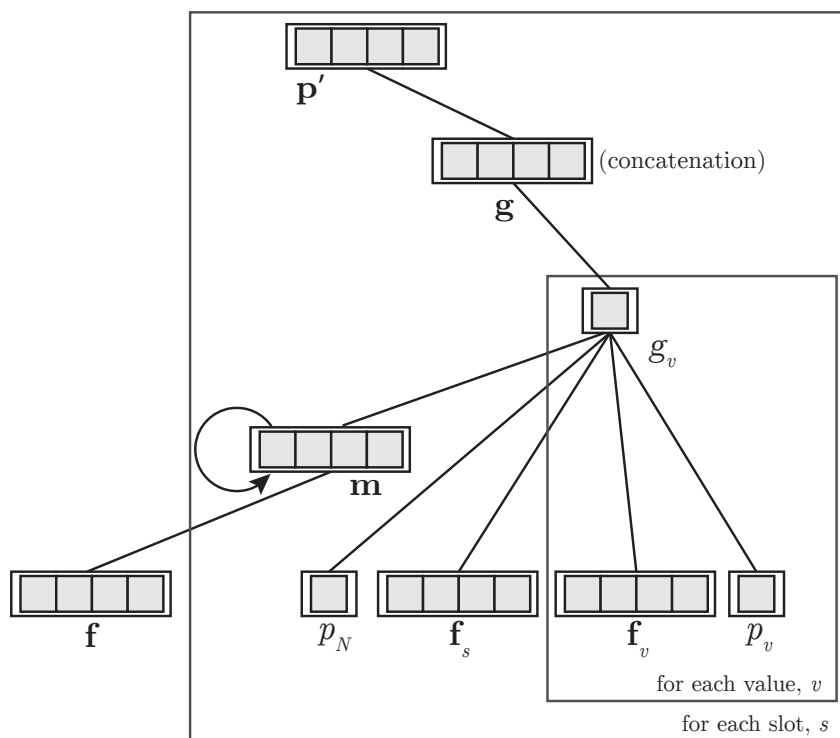


Figure 5.6: Delexicalised Recurrent Neural Network structure for DST. This is the same structure as given in equations (5.14-5.17), and figures 5.4 and 5.5, with the exception that the sub-network for \mathbf{h} is omitted.

By using regularisation (see appendix D), the learning will prefer where possible to use the sub-network for \mathbf{g} rather than learning the individual weights for each value required in the sub-network for \mathbf{h} . This sub-network is able to deal with unseen or infrequently seen dialogue states, so long as the state can be tagged in the feature extraction. This model can also be shared across informable slots since \mathbf{f}_s is included as an input, see section 5.7.2.

The sub-networks applied to tagged and untagged inputs are combined to give the new belief:

$$\mathbf{p}' = \text{softmax}([\mathbf{h} + \mathbf{g}] \oplus \{B\}) \in \mathbb{R}^{N+1} \quad (5.16)$$

where B is a parameter of the RNN, contributing to the *None* hypothesis (learnt as part of the Stochastic Gradient Descent (SGD) learning procedure). The definition of the softmax function is given in appendix D.

The contribution from \mathbf{g} may be seen as accounting for general behaviour of tagged hypotheses, while \mathbf{h} makes corrections due to correlations with untagged features and value specific behaviour e.g. special ways of expressing specific goals and fitting to specific SLU confusions.

Finally, the memory is updated as follows:

$$\mathbf{m}' = \sigma(W_{m_0}\mathbf{f} + W_{m_1}\mathbf{m}) \in \mathbb{R}^{N_{\text{mem}}} \quad (5.17)$$

where the W_{m_i} are parameters of the RNN, and $\sigma(x) = e^x / (1 + e^x)$.

5.5.4 Requested Slots and Search Method

A similar RNN structure is used to track the requested slots. Here the v runs over all the requestable slots $s \in S_{\text{req}}$, and requestable slot names are tagged in the feature vectors \mathbf{f}_v . This allows the neural network calculating \mathbf{g} to learn general patterns across slots just as in the case of goals. The equation for \mathbf{p}' is changed to:

$$\mathbf{p}' = \sigma(\mathbf{h} + \mathbf{g}) \quad (5.18)$$

(rather than the softmax) so each component of \mathbf{p}' represents the probability (between 0 and 1) of a slot being requested.

For search method classification, the same RNN structure as for a goal constraint is used. No tagging of the feature vectors is used in the case of search methods.

5.5.5 Key Parameters

There are several parameters of the RNN structure and SGD learning algorithm that must be specified. This section lists the key parameters, and gives default values for the parameters, which are used for the evaluations in this thesis unless it is specified otherwise in the text.

In learning, SGD is used to maximise the log probability of batches of $N = 20$ training dialogues at each step. An l2 regularisation term is used, weighted by $\lambda = 1.0$. Each learning step uses a fixed learning rate of $\eta = 0.1$. Gradients are clipped to lie between -1 and 1 . See appendix D for more information on the learning procedure.

The structure and size of the RNNs is specified by a few key dimensions. Firstly the size of the memory $|\mathbf{m}|$ is set to 5. The sub-network used to calculate \mathbf{h} has one hidden layer, of size $H_{\mathbf{h}}$ say, set to 32 for the RNNs used to track the search method and requested slots. For the RNN tracking the goal constraint for slot s , $H_{\mathbf{h}}$ is set to $|V_s| + 10$ (recall V_s is the set of possible values for s), i.e. the size of the hidden layer scales linearly with the size of the output layer for the sub-network. The sub-network used to calculate g_v has one hidden layer, of size $H_g = 32$.

5.6 Direct Word-based Tracking

Instead of using n -gram features extracted from the SLU as input to the RNNs, it is possible to directly use n -gram features extracted from the ASR output. Mapping directly from the ASR to the dialogue state is termed *word-based* DST. Doing so removes the need for an explicit SLU component, and directly estimates the dialogue state given the distribution over words output by the ASR. This results in high dimensional feature vectors (growing with the size of vocabulary), but neural networks are able to reduce high dimensional vectors into meaningful lower dimensional representations (Hinton and Salakhutdinov, 2006).

While tagging values in the SLU-derived n -grams to calculate \mathbf{f}_v is trivial, this needs some thought for ASR-derived n -grams. The simplest method is to do direct string matching on the ASR hypotheses, comparing the value in the database with the word string. For example “serving *italian*” is tagged to become “*serving* <value>”. This is a very similar problem to the task of finding alignments in SLU data discussed in section 3.2.1. A similar approach is adopted here, where a hand-written set of *aliases* is used to identify text realisations of slots and values in input sentences. Appendix C gives a full specification of the aliases used.

Relying on n -gram features may adversely affect the model’s ability to scale to new domains. Though n -gram features are shown to be sufficient in understanding the user’s speech

in the limited restaurant information domain (chapter 3), their limited representational powers may not be so effective in other domains. Unigrams, bigrams and trigrams may fail to represent adequately sentences in domains where complex grammar is used introducing e.g. long range dependencies between words.

However, the RNN framework may accept arbitrary features, and so more complex representations of the input could be considered for more complex domains. For example, it could be beneficial to exploit features extracted from grammatical parse trees, using paraphrase databases, and semantic relations from resources like Wordnet (Fellbaum, 1998). This is not studied in this thesis.

Word-based DST has two key features that make it an attractive method. The first is that it does not require any intermediate semantic format. This means that there is no need to design a dialogue act format such as given in appendix B. The model has the potential to outperform conventional DST, as it skips the inherent information bottle neck resulting from compressing the information in an SLU M -best list. By solving the problem in one step, there is no need to separately train an SLU on disjoint training data. The second advantage is that the RNN structure allows for pooling training data across slots, by removing the sub-network for \mathbf{h} . Such a network is termed a delexicalised word-based RNN, and is illustrated in figure 5.6. This means that models can be applied directly to new slots in new domains.

There is a special *Dontcare* value for each slot’s goal constraint, which means the user has specified they do not wish to constrain the slot. To allow modelling of this value, a special value-dependent feature is introduced $f_{v,i}$, which is 1 if $v = \text{Dontcare}$ and 0 otherwise. This allows the model to specifically learn the phrases that correspond to the *Dontcare* value.

Figure 5.7 demonstrates feature extraction for a word-based RNN tracker, a process analogous to SLU feature extraction described in section 5.5.1. The representation used is essentially the *weighted sum* representation of an ASR N -best list introduced in section 4.1.1. It would also be possible to use n -gram features derived from the *word confusion network*, but this is not investigated here.

5.7 Training Recurrent Neural Network Trackers

This section discusses training techniques for RNN-based DST. The training and development datasets from the second DSTC (*dstc2_train* and *dstc2_dev*) are used to evaluate the training techniques. For a full description of this data and the evaluation metrics, refer to chapter 6. The accuracy metric measures the correctness of the top dialogue state hypothesis, and the $L2$ metric measures the quality of the entire dialogue state distribution. A lower

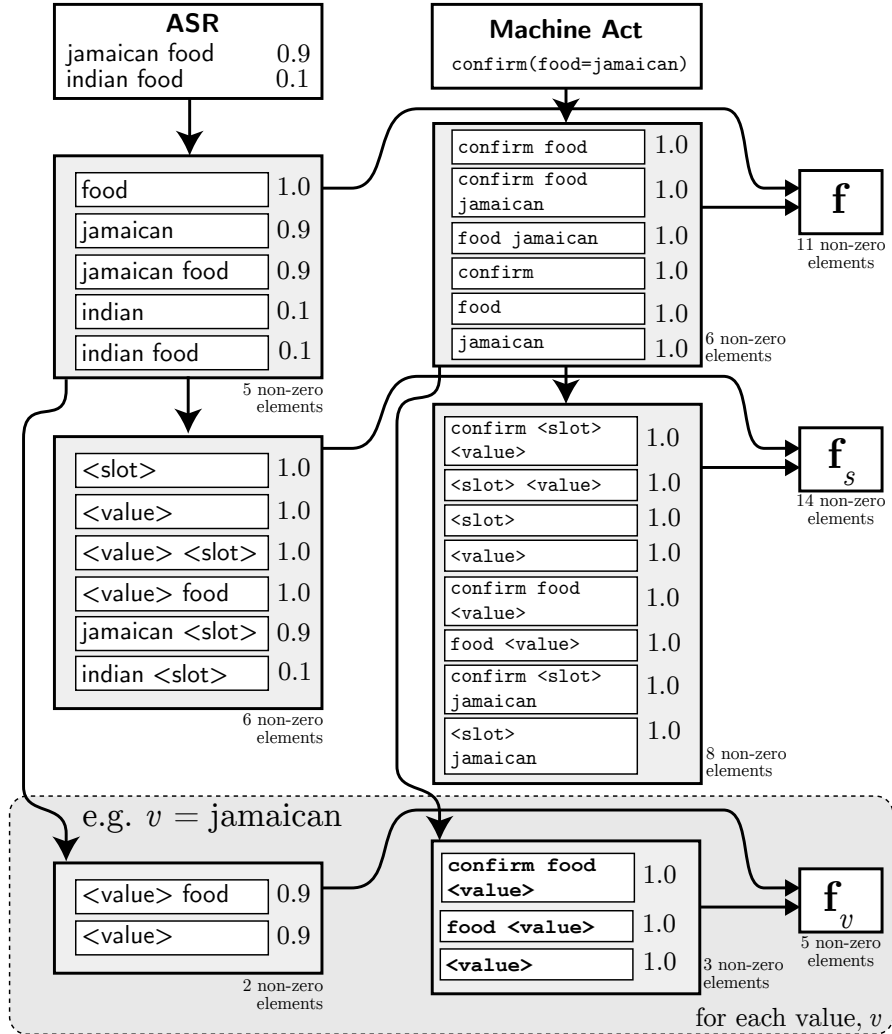


Figure 5.7: Example of feature extraction for one turn, giving **f**, **f_s** and **f_v**. Here *s* = *food*. For all $v \notin \{\text{indian, jamaican}\}$, **f_v** = **0**.

Shared init.	dA init.	Accuracy	L2
		0.686	0.477
	✓	0.688	0.466
✓		0.680	0.479
✓	✓	0.696	0.463
Baseline:		0.612	0.632

Table 5.2: Performance for the joint goal constraint subtask on the DSTC 2 dev set when varying initialisation techniques for word-based tracking. Full details of the dataset and evaluation metrics are given in section 6.1.2. For each row, 6 trackers are trained and then combined using score averaging. The final row shows the performance of the *focus* baseline tracker.

L2 score is better.

The Recurrent Neural Networks (RNNs) for DST are trained using back propagation through time and SGD, maximising the log probability of the sequences of observed beliefs in the training data for a fixed number of training epochs. Gradient clipping is used to avoid the problem of exploding gradients. A regularisation term is included, which penalises the l_2 norm of all the parameters. Further details are given in appendix D.

When using the ASR N -best list in word-based tracking, \mathbf{f} is typically of dimensionality around 4,000. With so many weights to learn, it is important to initialise the parameters well before starting the SGD algorithm. Two initialisation techniques have been investigated, the denoising autoencoder and shared initialisation. These were evaluated by training trackers on the *dstc2_train* set, and evaluating on *dstc2_dev* (see table 5.2).

5.7.1 Denoising Autoencoder Initialisation

A denoising Autoencoder (dA) is used to initialise the parameters of the RNN that multiply the high-dimensional input vector \mathbf{f} , following the technique presented in appendix D.

An l_1 regularisation term is added to the cross-entropy cost when learning the dA to encourage the learning of sparse weights. As the ASR features are likely to be very noisy, dense weights would be prone to overfitting the examples.

When initialising the weights in the RNN with weights learnt in a dA, training is observed to converge faster. Table 5.2 shows that dA initialisation leads to better solutions.

5.7.2 Shared Initialisation

It is possible to train a slot-independent delexicalised RNN, using training data from all slots, by not including \mathbf{h} in the model (the dimensionality of \mathbf{h} is dependent on the slot). In *shared initialisation*, such an RNN is trained for a few epochs, then the learnt parameters are used to initialise slot-dependent RNNs for each slot.

Table 5.2 suggests that performance is optimised by using a combination of shared initialisation with dA initialisation.

5.7.3 Model Combination

Model combination, combining the output of multiple RNNs trained with varying parameters, can help provide top accuracies. The technique for model combination used here is score averaging, where the final probability for each component of the dialogue state is computed as the mean of the probabilities output by all the trackers being combined. This is one of the simplest methods for model combination, and requires no extra training data. It is guaranteed to improve the accuracy if the outputs from the individual trackers are not correlated, and the individual trackers operate at an accuracy > 0.5 .

Multiple runs of training the RNNs were found to give results with high variability and model combination provides a method to exploit this variability. In order to demonstrate the effect, 10 trackers with varying regularisation parameters λ (see section 5.5.5) were trained on *dstc2_train* and used to track *dstc2_dev*. Figure 5.8 shows the effects of combining these trackers in larger groups. The mean accuracy in the joint goal constraints from combining m trackers is found to increase with m . The single output from combining all 10 trackers outperforms any single tracker in the group.

5.7.4 Training Set Size

Figure 5.9 presents an investigation into how much training data is needed to train the RNNs. The trackers are trained including the sub-network for \mathbf{h} . Performance on the test set is found to be stable after roughly 1,200 training dialogues have been provided.

5.8 Conclusions

This chapter has presented a variety of techniques for DST, including rule-based baseline methods, generative models, and discriminative models. Discriminative models are motivated by the fact that they can incorporate arbitrary features of the dialogue without making

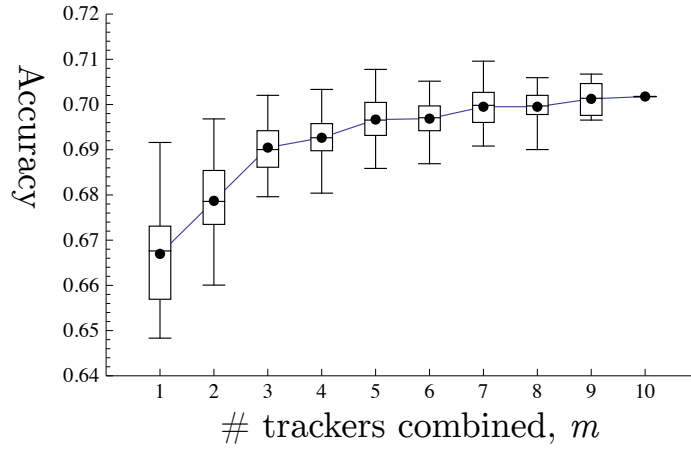


Figure 5.8: Joint goal constraint accuracy on *dstc2_dev* from system combination. Ten word-based RNN trackers are trained with varying regularisation parameters. For each $m = 1, \dots, 10$, all subsets of size m of the 10 trackers are used to generate $^{10}C_m$ (the binomial coefficient) combined results, which are plotted as a boxplot. Boxplots show minimum, maximum, the interquartile range and the median. The mean values are plotted as connected points.

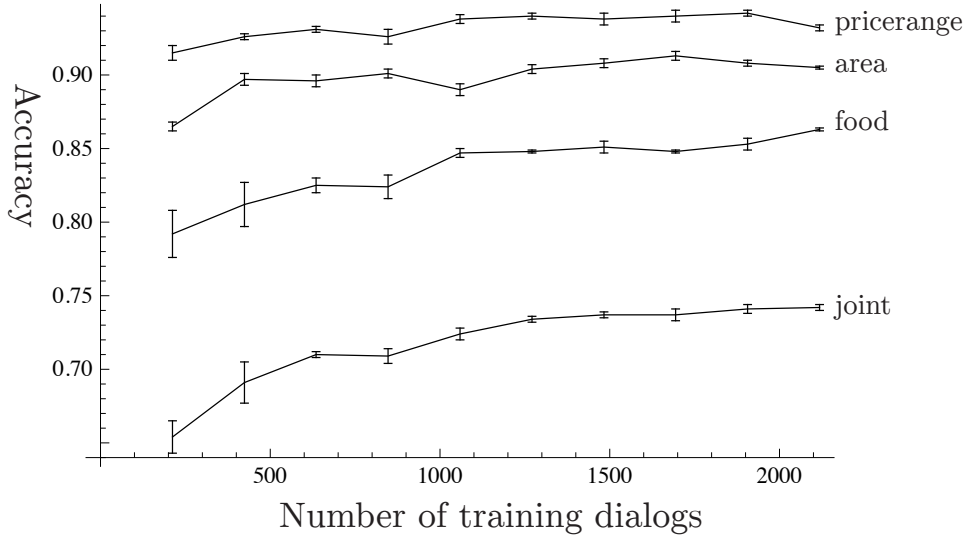


Figure 5.9: Goal constraint accuracy on the DSTC 2 test set for word-based RNN with varying amounts of training data. The mean and standard error of the accuracies of three trained RNNs are plotted.

any independence assumptions. An RNN-based framework is presented, which explicitly models the sequence of the dialogue while allowing for continuous high dimensional inputs.

Word-based tracking with RNNs is proposed, which avoids the need for any intermediate semantic interpretation, consolidating SLU and DST into one step that can be jointly optimised. By exploiting delexicalised inputs and pooling training data across slots, this technique should help in applying models to expanding domains (see section 6.3 and chapter 7).

The proposed RNN structure is factored over slots, classifying each slot's goal constraint separately. The dynamic Bayesian network is factored similarly, though any structure in the slots could be modelled in the network. Chapter 8 introduces an adaptation to the neural network structure to output a joint distribution over all slots concurrently.

The delexicalisation used here is simple string matching between the ASR hypotheses and the list of possible values and their aliases for a slot. In more complex domains or other languages this may not be sufficient, and other tagging processes should be investigated (some discussion of this is given in chapter 9).

The field of deep learning provides a plethora of structures for RNNs (such as long short-term memory gates (Hochreiter and Schmidhuber, 1997)) and techniques for learning the parameters of networks (such as ADAGRAD (Duchi et al., 2011)). Use of these may improve the performance of the learnt networks. In particular, an improved model of memory might be helpful for more complex domains, for example those where the users' language involves more dependencies with earlier turns.

Chapter 6 presents an evaluation framework for DST, and evaluates the methods presented here in an off-line corpus-based trial. Results of an online user trial will also be presented.

CHAPTER 6

EVALUATION OF DIALOGUE STATE TRACKING

Chapter 5 introduced a variety of methods for Dialogue State Tracking (DST), and this chapter seeks to evaluate them in an off-line corpus evaluation. A live user trial is also carried out to compare *discriminative* Recurrent Neural Network (RNN)-based trackers with a *generative* tracker.

Until recently, different research groups have used data from disparate domains, precluding direct comparisons of DST methods. The Dialog State Tracking Challenges (DSTCs) have provided common corpora and a standardised evaluation framework for DST (Williams et al., 2014). This has opened the problem to new researchers and smaller research groups in the field; developing a whole dialogue system, collecting the large number of dialogues required, and then labelling them is cumbersome, expensive, and time-consuming. As a result of the DSTCs evaluations, there is a considerable collection of competing methods with which RNN-based tracking can be directly compared, populating the rows of tables 6.2 and 6.3 in the evaluations presented here.

This chapter presents in detail the second and third DSTCs (DSTC 2 and 3), of which the author was the lead organiser. These are used to evaluate the methods for DST presented in chapter 5.

A live user trial is also presented in section 6.5, which is the first analysis of deploying discriminative DST in a live dialogue system.

6.1 Dialog State Tracking Challenges

As mentioned above, common dialogue corpora labelled with the dialogue state provide a useful resource for researchers in the field. The DSTCs have released such corpora and conducted blind evaluations on unlabelled test sets. The first Dialog State Tracking Challenge (DSTC), referred to as DSTC 1, released annotated dialogue corpora in the bus route information domain (Black et al., 2011; Williams et al., 2013). Here the dialogue state was composed of the user’s preferences for a variety of slots – route number, departure and arrival bus stops, time of day etc.

This section describes two follow-up challenges, DSTC 2 and DSTC 3. The author was the lead organiser for both of these challenges. The DST tasks in these challenges differ from DSTC 1 in several interesting ways. The most important is that it requires a more dynamic dialogue state; while in DSTC 1 the user goal was fixed and unchanging, users are allowed to change their mind in DSTC 2 and 3. DSTC 3 introduces the challenge of adapting models trained on one domain so that they may be applied to an expanded domain, i.e. a domain that has new slots and values.

The corpora for DSTC 2 and 3 were collected using subjects recruited via *Amazon Mechanical Turk*TM, and consist of dialogues in two domains: restaurant information, and tourist information (see appendix A). Tourist information subsumes restaurant information, introducing coffee shops and pubs. This expanded domain contains the same 9 slots as restaurant information (*type, area, food, name, pricerange, address, phone, postcode* and *signature*) but introduces new values for these slots, and also includes several new slots (*children allowed, has internet, has tv, near* and *price*). The two challenges were arranged as follows:

DSTC 2 released a large number of training dialogues in the restaurant information domain.

Compared to DSTC 1, DSTC 2 introduced changing user goals, tracking *requested slots, search methods* as well as the new restaurants domain. Results from the blind evaluation of DSTC 2 were presented in SIGdial 2014 (Henderson et al., 2014b).

DSTC 3 introduced the problem of adaptation to an expanded domain, tourist information.

DSTC 3 released only a small amount of labelled data (11 dialogues) in the tourist information domain; participants could then use this data plus the restaurant data from DSTC 2 for training, and trackers were evaluated on a large test set in the tourist information domain. Results from the blind evaluation of DSTC 3 were presented at SLT 2014 (Henderson et al., 2014d).

6.1.1 Data

A large corpus of dialogues with various telephone-based dialogue systems was collected using subjects recruited via *Amazon Mechanical Turk*TM, and divided into sets for the DSTC 2 and 3 research challenges.

DSTC 2

The DSTC 2 data consists of 3,235 dialogues in the restaurant information domain (see appendix A for descriptions of the dialogue domains). These dialogues were collected as part of a study comparing statistical and non-statistical approaches to dialogue management (Young et al., 2013). The paid *Amazon Mechanical Turk*TM crowd workers were assigned tasks and asked to call the dialogue systems. Callers were asked to find restaurants that matched particular constraints on the slots area, price range and food type. To elicit more complex dialogues, including changing goals, the users were sometimes asked to find more than one restaurant. In cases where a matching restaurant did not exist they were required to seek an alternative, for example finding an Indian instead of an Italian restaurant.

The dialogues come from 6 conditions; all combinations of 3 dialogue managers and 2 speech recognisers, with roughly 500 dialogues in each condition.

The 3 dialogue managers are:

DM-HC a simple tracker maintaining a single top dialogue state, and a hand-crafted *policy*

DM-POMDPHC a *dynamic Bayesian network* for tracking a distribution of dialogue states, and a hand-crafted policy

DM-POMDP the same tracking method as DM-POMDPHC, with a policy learnt using Partially Observable Markov Decision Process (POMDP) reinforcement learning

The 2 speech recognisers are:

ASR-degraded speech recogniser with artificially degraded statistical acoustic models

ASR-good full speech recogniser optimised for the domain

These give two acoustic conditions, the degraded model producing dialogues at higher Word Error Rates (WERs). The degraded models simulate in-car conditions and are described in Young et al. (2013).

When a tracker is deployed in an end-to-end dialogue system, it will inevitably alter the performance of the dialogue system it is part of, and change the distribution of dialogues

Slot	Dataset				
	dstc2_train	dstc2_dev	dstc2_test	dstc3_seed	dstc3_test
area	2.9%	1.4%	3.8%	0.0%	4.5%
food	37.3%	34.0%	40.9%	9.1%	8.0%
name	0.0%	0.0%	0.0%	0.0%	0.0%
pricerange	1.7%	1.6%	3.1%	9.1%	3.3%
children allowed	-	-	-	0.0%	0.2%
has tv	-	-	-	0.0%	0.1%
has internet	-	-	-	0.0%	0.3%
near	-	-	-	9.1%	0.8%
type	-	-	-	9.1%	2.6%
any	40.1%	37.0%	44.5%	27.3%	16.5%

Table 6.1: Percentage of dialogues that included a change in the goal constraint for each informable (and any slot) in the DSTC 2 and 3 data. Almost no users asked for venues by name.

relative to any previously collected. To simulate this, and to penalise over-fitting to known conditions, the test set (*dstc2_test*) contains dialogues using a dialogue manager that is not found in the training data. The *dstc2_test* set consists of all calls with DM-POMDP, in both speech recognition configurations. All calls with the other two dialogue managers are used for the training and development sets, *dstc2_train* and *dstc2_dev*. Specifically, the datasets are arranged as follows:

dstc2_train labelled dataset released in October 2013, with 1,612 dialogues from DM-HC and DM-POMDPHC, and both ASR conditions.

dstc2_dev labelled dataset released at the same time as *dstc2_train*, with 506 dialogues under the same conditions as *dstc2_train*. No caller in this set appears in *dstc2_train*.

dstc2_test set used for evaluation. Released unlabelled at the beginning of the evaluation week in January 2014. This consists of all 1,117 dialogues with DM-POMDP.

For one week in January 2014, the *dstc2_test* set was released without labels. At the end of the week, participants submitted their dialogue state tracker output. This allowed for a fair and blind evaluation of the trackers. The results were anonymised, but participants were allowed to identify their own submissions.

A breakdown of the frequency of goal constraint changes is given in table 6.1, showing around 40% of all dialogues involved a change in goal constraint. For each dataset

Dataset	Number of dialogues	Average number of turns per dialogue	Number of callers
dstc2_train	1,612	7.24	138
dstc2_dev	506	7.77	41
dstc2_test	1,117	8.85	174
dstc3_seed	11	9.09	11
dstc3_test	2,264	8.27	162

Table 6.2: Simple statistics of the DSTC 2 and 3 datasets.

Dataset	ASR	WER	F-score
dstc2_train	degraded	30.7%	72.4%
	good	22.4%	78.7%
	all	26.4%	75.7%
dstc2_dev	degraded	40.4%	67.3%
	good	25.2%	75.2%
	all	31.9%	71.6%
dstc2_test	degraded	33.6%	70.0%
	good	23.5%	77.8%
	all	28.7%	73.8%
dstc3_seed	-	19.3%	81.5%
dstc3_test	-	31.5%	78.1%

Table 6.3: WER and F-score statistics for DSTC 2 and 3 data. The F-score is calculated as in section 3.2.2. DSTC 2 datasets are split by the ASR condition (there is no ASR contrast in the DSTC 3 data).

and speech recogniser, table 6.3 gives the WER for the top Automatic Speech Recognition (ASR) hypothesis, and *F-score* for the top Spoken Language Understanding (SLU) hypothesis. The F-score and WER are calculated from the recorded performance of the live dialogue systems.

DSTC 3 Data

The DSTC 3 dialogues were collected using paid crowd-sourced workers, as part of a study into the Natural Actor and Belief Critic algorithms for parameter and policy learning in POMDP dialogue systems (Jurcicek et al., 2011). In total, 2,275 dialogues in the tourist information domain were collected. The dialogues are split into 2 sets as follows:

dstc3_seed A set of only 11 dialogues, released in April 2014 for debugging purposes.

dstc3_test A large test set of 2,264 dialogues, released unlabelled at the beginning of the evaluation week in June 2014.

As in DSTC 3, participants had one week with the unlabelled test data, at the end of which they were asked to submit their dialogue state tracker output for evaluation. Participants could exploit the *dstc2_train*, *dstc2_dev* and *dstc2_test* datasets (all fully labelled at this point) in the smaller restaurant information domain when developing their trackers for the tourist information domain in DSTC 3. Recall that the tourist information domain subsumes the restaurant information domain, adding 5 new slots (see appendix A).

One key mis-match between the DSTC 2 and 3 datasets is the frequency of goal changes in the data. Table 6.1 shows that 16.5% of the *dstc3_test* dialogues contained a change of goal, which is less than half of the percentage in the DSTC 2 data. Table 6.3 gives the WER and F-score statistics calculated from the live performance of the recorded dialogues for the DSTC 2 and 3 data. The ASR performance was similar to the DSTC 2 data, but the SLU performed slightly more accurately.

Labelling

The dialogue state labels were collected by first labelling each user utterance with its semantic representation, in the dialogue act format described in appendix B (some example semantic representations are shown in figure 5.1). The semantic labelling was achieved by first crowd-sourcing the transcription of the audio to text. Next a semantic decoder was run over the transcriptions, and the author corrected the decoder’s results by hand.

Given the sequence of machine actions and labelled user actions, both represented semantically, the true dialogue state is computed deterministically using a simple set of rules.

6.1.2 Evaluation Metrics

The full results of the DSTC 2 and 3 evaluations include a large bank of metrics, which measure performance in multiple dimensions for different parts of the dialogue state. In DSTC 2, there are a total of 815 metrics calculated per tracker. Although each metric has its own particular motivation, many of the metrics are highly correlated. From the results of DSTC 1 it was found the metrics could be roughly split into 3 independent groups – one measuring 1-best quality, another measuring probability calibration, and the last measuring discrimination (Williams et al., 2013).

This motivates the official *featured metrics* of DSTC 2 and 3 – *accuracy* to measure the 1-best quality and *L2* to measure the probability calibration. The third group mentioned above, of metrics measuring discrimination, is of less interest – as it only makes sense to compare trackers based on their discrimination if they are competing at similar accuracies (Henderson et al., 2014b). Participants in the challenge competed to perform top for these featured metrics, and these metrics will be used to report DST performance in the remainder of this chapter.

The accuracy metric measures the fraction of *turns* where the top hypothesis for a component of the dialogue state is correct. The L2 metric is the square of the L2 norm between the distribution reported by a tracker and the correct label, indicating quality of the whole reported distribution. It is calculated for one turn as:

$$L2 = (1 - p_i)^2 + \sum_{j \neq i} p_j^2 \quad (6.1)$$

where \mathbf{p} is the reported distribution, and p_i is the probability assigned to the correct hypothesis. Note a lower L2 score is considered better.

The accuracy and L2 metrics are reported for the joint *goal constraints*, search method and requested slots – giving 6 metrics in total.

The metrics are calculated only on turns where there is some information about the dialogue state component in question from the SLU or system output so far (*schedule 2* in the terminology of the DSTCs). For example, the joint goal constraints metrics are calculated for all turns in a dialogue after a constraint has appeared in an SLU hypothesis or is part of the system’s output dialogue act.

6.1.3 Oracle Tracker

The results of an *oracle* tracker, with access to the labels, are included in these evaluations. The oracle tracker reports the correct label with score 1 for each component of the dialogue state, but only if it has been suggested in the dialogue so far by the SLU (only if the corresponding SLU observation o_i is non-zero at some point in the dialogue history).

The performance of this tracker gives an upper-bound for the DST performance of trackers using only the hypotheses suggested by the SLU. Note that the RNN trackers that include the \mathbf{h} component have the capacity to beat this oracle in terms of accuracy in at least some turns, as they can output hypotheses that are not included in the SLU output.

6.2 The Second Dialog State Tracking Challenge Evaluation

Nine research teams, one of which was the author, participated in the strict blind evaluation of the DSTC 2 challenge. In total 31 trackers were evaluated. This blind evaluation procedure was used to investigate the performance of the RNN approach to DST under two contrasts:

- **Input features** – contrasting *word-based* DST (using ASR features), with conventional tracking based on SLU features.
- **Inclusion of \mathbf{h}** – contrasting including and omitting the sub-network for \mathbf{h} in the RNN. Recall \mathbf{h} is the part of the model that allows learning special behaviours for particular dialogue state hypotheses, and correlations with untagged features (see figures 5.4 and 5.5).

These two binary contrasts resulted in a total of 4 system variants being entered in the challenge. Two further variants were evaluated after the official evaluation of the challenge, using combined ASR and SLU features with and without \mathbf{h} .

For each variant, the final tracker is the averaged output of 12 RNN trackers trained with varying parameters (see section 5.7.3). The regularisation weight λ (see section 5.5.5) is set at 0.1 or 1. For models using \mathbf{h} , the parameters corresponding to the sub-network for \mathbf{h} are weighted in the regularisation term by a factor of 0.1, 1 or 10. All combinations of the two values for λ and the 3 values for this factor give 6 parameter settings, each of which is used to train 2 RNNs trackers for the final ensemble of 12 trackers. For models that do not use \mathbf{h} , the parameter H_g (see section 5.5.5) is set at 10, 32 or 40, in place of varying the regularisation factor.

In this evaluation, no *aliases* are used to aid in delexicalisation (see appendix C). Instead, delexicalisation simply looks for exact string matches of the slot values and slot names as they appear in the domain specification.

The results of the evaluation are given in table 6.4. Full results and further analysis can be found in Henderson et al. (2014b).

It should be noted that the live SLU used the *word confusion network*, not made available in the challenge. The word confusion network is known to provide stronger features than the *N-best list* for language understanding (Henderson et al. (2012); Tur et al. (2013); section 3.1.3), so the word-based trackers using *N-best list* ASR features were at a disadvantage in that regard. Nevertheless, despite this handicap, the best results were obtained from

6.2. The Second Dialog State Tracking Challenge Evaluation

	Features		Joint Goals		Search Method		Requested	
	ASR	SLU	Acc.	L2	Acc.	L2	Acc.	L2
RNN, with h	✓		0.768	0.346	0.940	0.095	0.978	0.035
RNN, no h	✓		0.746	0.381	0.939	0.097	0.977	0.038
team2 entry2 ¹	✓		0.668	0.505	0.944	0.095	0.972	0.043
team7 entry0 ²	✓		0.750	0.416	0.936	0.105	0.970	0.056
RNN, with h		✓	0.742	0.387	0.922	0.124	0.957	0.069
RNN, no h		✓	0.737	0.406	0.922	0.124	0.957	0.069
Bayesian net.		✓	0.675	0.550	0.880	0.210	0.885	0.197
1-best baseline		✓	0.619	0.738	0.879	0.209	0.884	0.196
focus baseline		✓	0.719	0.464	0.867	0.210	0.879	0.206
HWU baseline ³		✓	0.711	0.466	0.897	0.158	0.884	0.201
team1 entry0 ⁴		✓	0.601	0.648	0.904	0.155	0.960	0.073
team3 entry0 ⁵		✓	0.729	0.452	0.878	0.210	0.889	0.188
team6 entry2		✓	0.718	0.437	0.871	0.210	0.951	0.085
team7 entry4 ²		✓	0.735	0.433	0.910	0.140	0.946	0.089
team8 entry1 ⁶		✓	0.699	0.498	0.899	0.153	0.939	0.101
team9 entry0		✓	0.499	0.760	0.857	0.229	0.905	0.149
*RNN, with h	✓	✓	0.770	0.341	0.941	0.091	0.978	0.036
*RNN, no h	✓	✓	0.762	0.366	0.941	0.092	0.976	0.039
team2 entry1 ¹	✓	✓	0.784	0.735	0.947	0.087	0.957	0.068
team2 entry3 ¹	✓	✓	0.771	0.354	0.947	0.087	0.941	0.090
team5 entry4	✓	✓	0.695	0.610	0.927	0.147	0.974	0.053
SLU-based oracle			0.850	0.300	0.986	0.028	0.957	0.086

Table 6.4: Results of DSTC 2 evaluation, showing performance of RNN trackers alongside other teams (¹Williams (2014), ²Sun et al. (2014b), ³Wang and Lemon (2013), ⁴Kim and Banchs (2014), ⁵Smith (2014) and ⁶Lee et al. (2014)). The top performing trackers from each team are selected. Results are split by the input features used, with bold indicating the top result in the group. The RNN trackers were submitted under team4 (Henderson et al., 2014a). An asterisk (*) indicates results from systems not submitted to the DSTC evaluation.

word-based tracking directly on the ASR output, rather than using the word confusion network generated SLU output. Including **h** always helps, though this is far more pronounced for the word-based trackers.

The RNN trackers performed very competitively in the context of the challenge. The word-based tracker including **h** (**h**-ASR), was top for joint goal constraints L2 as well as requested slots accuracy and L2 among all trackers in DSTC 2. It was close to the top for the other featured metrics, following closely entries from team 2 (Williams, 2014), which used a static classifier approach (section 5.4.1). Williams used a discriminative ranker, which performed well in terms of accuracy, but was among the worst for the L2 metric. The ranker learns to put the correct hypothesis near the top in many cases, but it is not trained to output well-calibrated probability scores. The quality of the probability scores is important for statistical dialogue systems.

There are hundreds of metrics reported in the DSTC 2 and 3 evaluations, and it was found that the **h**-ASR tracker ranked top on many of them. Considering L2, accuracy, average probability, equal error rate, log probability and mean reciprocal rank across all components of the dialogue state, these give a total of 318 metrics. The **h**-ASR tracker ranked top of all trackers in the challenge in 89 of these metrics, more than any other tracker. The ASR tracker omitting **h** came second, ranking top in 33 of these metrics.

The RNN trackers using SLU features ranked top in all of the featured metrics among the trackers that used only the SLU output. Using both ASR and SLU features gave a modest improvement in performance for tracking the goal constraints and search method.

Note that the word-based RNN trackers were able to improve on the SLU-based oracle for tracking the requested slots, as they were able to propose suggestions not found in the SLU output.

The generative technique using a dynamic Bayesian network (section 5.3) performs relatively poorly in the evaluation. Though it does better than the one-best baseline, it performs worse than the other baseline methods in tracking the goal constraints.

6.3 The Third Dialog State Tracking Challenge Evaluation

Recall the third DSTC studied the ability of trackers to adapt to an expanded domain. Training data was available in the smaller domain of DSTC 2 concerning restaurant information, while the test data included coffee shops and pubs as well as restaurants. The set of possible values for each slot was different in the test set, which also included 5 new slots. A summary of the difference between the smaller training domain and expanded test domain of DSTC 3 is given in appendix A.

As with DSTC 2, variant RNN trackers were submitted for the blind evaluation of DSTC 3. Seven research teams contributed a total of 23 trackers to the evaluation. The submitted RNN trackers varied in the features they used. The first was a word-based tracker using ASR features (ASR) and the second used both ASR and SLU features (ASR+SLU). After the official evaluation, a tracker using only SLU features (SLU) was evaluated using the same procedure.

For the challenge of tracking an expanded domain, it is useful to exploit *delexicalised* RNNs trackers, i.e. those that do not include the sub-network for \mathbf{h} . As discussed in section 5.7.2, such trackers are capable of learning across slots and can be applied to new slots and values not found in the training data.

In all cases slot-independent delexicalised RNNs, which did not include the sub-network for \mathbf{h} , were first trained on the labelled data for all slots in the training set. The slot-independent models were used to track the new slots in the expanded test domain. For slots existing in the training set, slot-dependent models were trained by starting from the slot-independent model and continuing training using only the labelled data for that slot.

For each tracker, an ensemble of six separate RNNs were trained with varying parameters, and then combined using score averaging. The six variants arose from all combinations of H_g (the size of the hidden layer for the sub-network for g_v) being set at 32, 50 or 100 and the memory size $|\mathbf{m}|$ being set at 5 or 10 (see section 5.5.5). The word-based delexicalised RNN trackers used a set of aliases to help in the delexicalisation process. Appendix C provides a full specification of the aliases used.

Note that word-based systems using only ASR features are attractive as they do not require an SLU system for the new domain. A tracker using SLU features leaves open the question of how to design SLU systems for expanding domains, while word-based trackers avoid the need for this and require no intermediate semantic representations or taxonomy of dialogue acts.

The performance in the DSTC 3 evaluation is presented in table 6.5. The full evaluation and further analyses are given in Henderson et al. (2014d).

The delexicalised RNN trackers performed consistently well across all tasks and evaluation metrics. The ASR+SLU tracker obtained the top accuracy and L2 scores for tracking the joint goal constraints in the challenge. Among the trackers that did not use the SLU, the word-based ASR tracker obtained the top joint goal constraint accuracy. The SLU tracker also performed strongly, particularly in the quality of its probability scores as measured by the L2 metrics. Higher accuracies in this setting were obtained in the Conditional Random Field (CRF)-based model of team7 (Ren et al., 2014b).

Note that the *Bayesian network* tracker performed more competitively than in the DSTC

	Features		Joint Goals		Search Method		Requested	
	ASR	SLU	Acc.	L2	Acc.	L2	Acc.	L2
RNN	✓		0.616	0.565	0.966	0.061	0.939	0.100
team5 entry0 ¹	✓		0.610	0.556	0.968	0.091	0.949	0.090
*RNN		✓	0.570	0.611	0.965	0.062	0.938	0.104
Bayesian net.		✓	0.565	0.741	0.923	0.153	0.778	0.394
1-best baseline		✓	0.555	0.860	0.922	0.154	0.778	0.393
focus baseline		✓	0.556	0.750	0.908	0.134	0.761	0.435
HWU baseline ⁴		✓	0.575	0.744	0.967	0.062	0.767	0.417
team1 entry3		✓	0.561	0.733	0.963	0.097	0.774	0.401
team6 entry0		✓	0.507	0.736	0.927	0.120	0.907	0.157
team7 entry1 ²		✓	0.576	0.652	0.957	0.116	0.938	0.101
RNN	✓	✓	0.646	0.534	0.966	0.061	0.943	0.091
team2 entry0	✓	✓	0.585	0.697	0.965	0.114	0.929	0.121
team2 entry3	✓	✓	0.582	0.639	0.970	0.065	0.938	0.138
team4 entry0 ³	✓	✓	0.630	0.627	0.853	0.272	0.923	0.136
SLU-based oracle			0.717	0.565	0.988	0.02	0.946	0.107

Table 6.5: Results of DSTC 3 evaluation, showing performance of RNN trackers alongside other teams (¹Sun et al. (2014a), ²Ren et al. (2014b), ³Kadlec et al. (2014b) and ⁴Wang and Lemon (2013)). The top performing trackers from each team are selected. Results are split by the input features used, with bold indicating the top result in the group. The RNN trackers were submitted under team3 (Henderson et al., 2014c). An asterisk (*) indicates results from systems not submitted to the DSTC evaluation.

	Accuracy	L2
<i>All dialogues</i>		
Bayesian network	0.630	0.618
focus baseline	0.665	0.563
SLU-based oracle	0.801	0.398
<i>No change in goal</i>		
Bayesian network	0.694	0.519
focus baseline	0.681	0.541
SLU-based oracle	0.824	0.351
<i>At least one change in goal</i>		
Bayesian network	0.523	0.785
focus baseline	0.642	0.600
SLU-based oracle	0.733	0.453

Table 6.6: Breakdown of performance on joint goal constraint accuracy and L2 for dialogues with and without at least one change in the goal constraints. There are 1,627 dialogues with at least one change in the goal constraints, and 2,750 with no change.

2 evaluation, outperforming the baseline trackers across all metrics. The difference in performance is thought to be due to the much lower percentage of changing goal constraints in the DSTC 3 data (see table 6.1). This is explored further in section 6.4.

6.4 Changing Goal Constraints

This section investigates the hypothesis that the focus baseline is outperforming the Bayesian network tracker specifically in dialogues with changing user goal constraints. A collection of 4,377 dialogues in the restaurant information domain (see appendix A), which contains as a subset the DSTC 2 data, was split into two subsets – those containing a change in the true goal constraint for some slot, and those for which the goal constraint does not change.

Table 6.6 shows the performance on these two subsets for the two trackers. Results of the SLU-based oracle are also given. The oracle tracker performs more accurately in the dialogues with no change in goal, suggesting these dialogues are inherently easier to track than the dialogues containing changes of goal. The table confirms that while the generative Bayesian network tracker does slightly better in dialogues with no change in goal, the focus baseline is able to deal with goal changes far better.

Table 6.7 presents a similar analysis on the DSTC 2 test set, which is a smaller set

consisting of 1,117 dialogues. Using this set allows for the inclusion of trackers from the DSTC 2 evaluation (see section 6.2). Three discriminative trackers are included, two RNN-based trackers and the ranking method of Williams (2014), which achieved the top joint goal constraint accuracy in the evaluation. The results again show that the performance of the Bayesian network tracker degrades substantially in dialogues containing a change in goals. On the other hand, the discriminative trackers and focus baseline tracker do not degrade to the same extent.

	Accuracy	L2
<i>All dialogues</i>		
Bayesian network	0.675	0.550
focus baseline	0.719	0.464
RNN with SLU input	0.742	0.387
RNN with ASR input	0.768	0.346
(Williams, 2014)	0.784	0.735
SLU-based oracle	0.850	0.300
<i>No change in goal</i>		
Bayesian network	0.751	0.428
focus baseline	0.753	0.411
RNN with SLU input	0.761	0.354
RNN with ASR input	0.788	0.319
(Williams, 2014)	0.807	0.711
SLU-based oracle	0.878	0.244
<i>At least one change in goal</i>		
Bayesian network	0.607	0.657
focus baseline	0.689	0.511
RNN with SLU input	0.725	0.416
RNN with ASR input	0.751	0.371
(Williams, 2014)	0.764	0.756
SLU-based oracle	0.826	0.349

Table 6.7: Breakdown of performance on joint goal constraint accuracy and L2 for dialogues with and without at least one change in the goal constraints, for the DSTC 2 test set. There are 497 dialogues with at least one change in the goal constraints, and 620 with no change. The two RNN-based trackers are the trackers evaluated in section 6.2 that include the sub-network for **h**. Also included is the discriminative ranking approach of Williams (2014), chosen as it achieved the top goal constraint accuracy in the DSTC 2 evaluation.

To demonstrate the problem, an example dialogue is given in figure 6.1. The focus base-

line is able to forget quickly the previous hypothesis for the food goal, while the generative model needs multiple *informs* for the goal to change to the new correct value. This is due to the parameter for how likely a goal is to change from one turn to the next. If this probability were higher, then the goal would change quicker, but then the distribution would flatten out as the dialogue progresses and the goal is not mentioned. When modelling the evolution of goals in a generative model it is difficult to simultaneously model goal changes and permit long lasting memory of the goal from one turn to the next.

This issue has been addressed to some extent in recent work on improving generative models for DST. Kadlec et al. (2014b) introduce a parameter called the *durability* to explicitly model and control how easily users change goals through the evolution of the dialogue. Including this parameter into a generative Bayesian network model improves tracking performance to be comparable to the baselines and machine-learned models that use SLU input. However discriminative models that are trained on ASR input (such as the word-based RNN tracker presented in this section) still outperform this approach.

6.5 Live User Evaluation

Discriminative Dialogue State Tracking (DST) with RNNs performed strongly in the off-line evaluations of the second and third Dialog State Tracking Challenges (DSTCs), obtaining top scores in many of the metrics. In general, the DSTCs have brought DST into focus in recent years, and many research teams have made improvements over the previous state of the art as measured by the metrics on off-line corpus-based evaluations.

It is important to assess the performance in an end-to-end dialogue system. No published result to date confirms the expected hypothesis that improvements on the off-line metrics translate to an improvement in overall dialogue quality when deployed in an end-to-end system¹. Deploying a new dialogue state tracker may affect performance in an unpredicted manner, e.g. by skewing the distribution of dialogues seen by the system relative to the training corpora. It might also be possible for a tracker to do well on the off-line metrics that are calculated as averages over many dialogues and turns, while not doing well enough in specific key situations.

The proposed RNN-based method for DST is evaluated in an end-to-end dialogue system. Gaussian process reinforcement learning is used to first learn a policy (mapping from dialogue state to actions) for each tracker in the evaluation, in interaction with real users. The resulting policies are then evaluated in further interaction with callers.

¹A trial using simulated users presented by Kim et al. suggests that improvements in the off-line metrics should be expected to translate to improved dialogues.

1	<i>Hello, welcome to the Cambridge restaurant system. You can ask for restaurants by area, price range or food type. How may I help you?</i>	SLU 0.95 <i>inform(food=unusual)</i> 0.05 <i>inform(pricerange=cheap)</i> 0.05 <i>inform(food=unusual)</i>	Focus 0.95 food=unusual ✓
		True semantics <i>inform(food=unusual, pricerange=cheap)</i>	Bayesian network 0.95 food=unusual ✓
2	<i>I'm sorry but there is no restaurant serving unusual food.</i>	SLU 0.66 <i>regalts(food=european)</i> 0.16 <i>inform(food=european)</i> 0.07 <i>regalts()</i> 0.04 <i>affirm(food=european)</i> 0.03 <i>hello(food=european)</i> 0.02 <i>confirm(food=european)</i> 0.02 <i>negate(food=european)</i>	Focus 0.01 food=unusual ✓ 0.99 food=european ✓
		True semantics <i>regalts(food=european)</i>	Bayesian network 0.64 food=unusual × 0.32 food=european ×
3	<i>I'm sorry but there is no restaurant serving unusual food.</i>	SLU 0.26 <i>regalts()</i> 0.06 <i>ack()</i> 0.05 <i>repeat()</i> 0.04 <i>affirm()</i> 0.04 <i>thankyou()</i> 0.03 <i>negate()</i> 0.02 <i>hello()</i>	Focus 0.01 food=unusual ✓ 0.99 food=european ✓
		True semantics <i>regalts(food=european)</i>	Bayesian network 0.64 food=unusual × 0.32 food=european ×
4	<i>I'm sorry but there is no restaurant serving unusual food.</i>	SLU 0.91 <i>inform(food=european)</i> 0.02 <i>deny(food=european)</i> 0.02 <i>regalts(food=european)</i> 0.02 <i>confirm(food=european)</i> 0.01 <i>affirm(food=european)</i>	Focus 0.01 food=unusual ✓ 0.99 food=european ✓
		True semantics <i>inform(food=european)</i>	Bayesian network 0.01 food=unusual ✓ 0.99 food=european ✓

Figure 6.1: Example dialogue, taken from DSTC 2 data, illustrating a change in goal constraint. This shows the output of the focus baseline and Bayesian network trackers for the food slot. The remaining sum of the SLU M -best list is assigned to *null()* hypothesis. A tick or cross indicates whether the top hypothesis is correct. In this example the focus baseline gets an accuracy of 1.0 while the generative Bayesian network tracker gets 0.5.

The results of the live evaluation show that a word-based RNN tracker performs top in every measurement of dialogue quality, outperforming a generative Bayesian network tracker and another RNN tracker that relies on the output of an SLU component.

Section 6.5.1 describes the setup of the experiment, explains the online policy learning procedure, and presents the evaluation metrics. Section 6.5.2 presents and analyses the results.

6.5.1 Experimental Setup

Participants were employed using *Amazon Mechanical Turk*TM to interact with multiple dialogue systems in the restaurant information domain. When calling, participants would be redirected randomly to one of 6 variant dialogue systems, arising from all combinations of 3 dialogue state trackers and 2 acoustic conditions.

The three dialogue state trackers evaluated are:

- **BN** – a Bayesian network tracker, i.e. the state of the art before the DSTCs (described in section 5.3). This uses the *CNet decoder* (section 3.1.3) for Spoken Language Understanding (SLU).
- **RNNSLU** – an RNN tracker that uses SLU features from the CNet decoder.
- **RNNASR** – a word-based RNN tracker that works directly on the ASR *N*-best list, without using any semantic decoder.

These are the dialogue state trackers exactly as evaluated in the DSTC 2 evaluation (section 6.2). The two RNN trackers both include the sub-network for **h**, i.e. this is not evaluating delexicalised DST. Recall in this off-line evaluation, the RNNASR tracker outperformed the other two trackers, and came top in most of the metrics in the challenge.

Each of the three trackers is evaluated in two acoustic conditions:

- **Full ASR** – using speech recognition models fully trained on in-domain data.
- **Degraded ASR** – using under-trained speech recognition models, which closely simulate the effect of poor acoustic conditions.

These are the two acoustic conditions used to generate the DSTC 2 data (see section 6.1.1). Recall the degraded models result in dialogues at higher WERs, simulating poor acoustic conditions similar to those in a moving automobile (Young et al., 2013).

The participants are given tasks such as “*Try to find a Chinese restaurant in the West of town, if there is no such place try any part of town. Make sure you get the address and*

phone number.” These tasks included goal changes, and for the user to sometimes ask for alternative suggestions.

All the variant dialogue systems use a POMDP as with the Bayesian Update of Dialogue State (BUDS) system (Thomson and Young, 2010) so that a dialogue policy can be learnt. However the policy, the mapping from the dialogue state tracker’s output to the next system action (see section 2.4), should be adapted to the acoustic conditions and dialogue state tracker of the dialogue system. Any fixed policy could arbitrarily be better suited for some of the dialogue state tracker or acoustic conditions.

Online policy learning provides an ideal solution for learning a suitable policy in any given dialogue state tracker and acoustic condition.

Online Policy Learning

A dialogue policy can be optimised in direct interaction with users using online policy learning. The approach taken here is to use Gaussian process reinforcement learning, which internally uses Gaussian processes to model the Q function, a key function in reinforcement learning. Recall from section 2.4, the Q function $Q(s, a)$ is a function of the dialogue state s and each possible next system action a , and is an estimate of the expected future cumulative *dialogue reward* if the action a is to be taken in state s .

Gaussian process reinforcement learning has recently been shown to provide a framework for efficiently learning an optimal policy for a statistical Spoken Dialogue System (SDS). The method learns a policy that outperforms those learnt in simulated interactions, while requiring orders of magnitude fewer dialogues (Gašić et al., 2013).

Fast learning is facilitated by the use of a robust dialogue reward, which compares subjective and objective measures of the dialogues. At the end of each dialogue, the user is asked whether or not they found the information they were looking for – answering ‘yes’ provides a subjective success of 1 and ‘no’ a subjective success of 0. As the task given to the user is known to the system, an objective measure of success can also be calculated. If the user is able to find a venue matching the given constraints, and obtain all the required information from the system, then the objective success is 1, otherwise it is 0. Dialogues for which the subjective and objective success measures are not equal are then filtered, and do not take part in the policy learning. For dialogues where the success measures are equal, the reward R is used:

$$R = 20 \cdot \text{success} - \text{Number of user turns} \quad (6.2)$$

which favours successful yet concise dialogues. Dialogues for which the two success mea-

tures agree, and which are used in policy learning, are termed *learning dialogues*.

Gašić et al. find that an optimal policy can be learnt using Gaussian process reinforcement learning after 1,200 such learning dialogues. The evaluation of each of the 6 variant dialogue systems is done in two phases:

- **Online policy learning phase** - each of the 6 systems (3 trackers and 2 acoustic conditions) is initiated with a random policy, and Gaussian process reinforcement learning is used to learn a policy in interaction with real users. In total 1,200 dialogues are used to train each policy.
- **Learnt policy evaluation phase** - After policy learning, the policies are frozen and the 6 systems take part in roughly another 500 dialogues each.

Each of these two phases is carried out concurrently for all 6 systems. In total, 10,169 dialogues are collected.

Evaluation Metrics

A variety of metrics are reported in the evaluation phase.

As mentioned above, after each dialogue the user declares whether they thought the dialogue was successful. This gives the *subjective reward* which is 1 if successful or 0 otherwise.

Recall the automatically calculated *objective success* metric is 1 if the user is able to find a venue matching the given constraints and obtain all the required information from the system, otherwise it is 0.

Two dialogue reward scores are included in the evaluation. The first is called the *Subjective reward*, and is the same metric included in the live evaluation of section 4.3. This is calculated as:

$$\text{Subjective reward} = 20 \cdot (\text{subjective success}) - \text{Number of turns} \quad (6.3)$$

The above metrics are reported as averages over all of the evaluation dialogues. So for example the binary success metrics become success rates, given as percentages. The second dialogue reward score reported is the *filtered reward*. This score is identical to the subjective reward, except it is calculated only over evaluation dialogues for which the two success metrics are equal. The filtered reward is the metric that the online policy learning is designed to optimise.

The participants are also asked a question “*The system understood me well,*” with 5 answers ranging from “*strongly disagree*” to “*strongly agree*”. This gives a metric ranging from 0-4, which is also presented in the results.

6.5.2 Results

Table 6.8 presents the results from the evaluation dialogues. Some statistical significance results are given in table 6.9, which pools the dialogue state trackers across acoustic conditions in order to give fewer comparisons. Figure 6.2 gives learning curves for the 6 systems during the online policy learning phase.

In general it seems that the RNN methods for DST do indeed improve end-to-end dialogue system performance. In table 6.9, all of the metrics place the dialogue state trackers in the order of RNNASR, RNNSLU then BN from best to worse, and almost all of these comparisons are statistically significant. This agrees with the ordering of the off-line evaluation.

The RNN trackers show strong robustness to the noise of the degraded ASR condition. The subjective and objective success rates of the RNNASR system in the degraded conditions are higher than those obtained by the BN system with the full acoustic models. The relative difference between the BN and RNNASR subjective success rates is quite large, almost 10% absolute in both acoustic conditions.

Recall the dialogue policies were trained to maximise the filtered reward. For this metric the RNN trackers significantly outperform the Bayesian network tracker. The system with the word-based RNNASR tracker is able to score around 3 more points in the dialogue reward than the BN system. This is a substantial improvement compared to the gain of 0.5 points found by using a more accurate SLU component in section 4.3.

In conclusion, the live evaluation has found significant and substantial gains in employing the RNN based dialogue state trackers in every metric evaluating end-to-end dialogue quality. The off-line evaluation results were reflected in the ordering of the systems, with the word-based RNN performing particularly strongly.

6.6 Conclusions

The proposed RNN-based framework for DST performed competitively in the off-line evaluations. Word-based tracking performed particularly well, a configuration that avoids the need for any intermediate semantic interpretation, consolidating SLU and DST into one step that can be jointly optimised. This model was found to produce high quality dialogue state



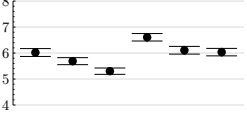
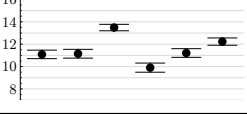
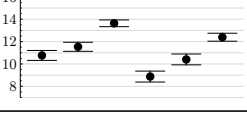
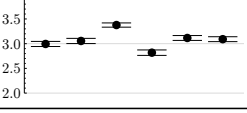
		ASR condition		
		Full	Degraded	
Subjective success rate (%)	BN	85.5 \pm 1.6	82.6 \pm 1.7	
	RNNSLU	84.2 \pm 1.6	86.6 \pm 1.5	
	RNNASR	94.0 \pm 1.1	91.3 \pm 1.3	
Objective success rate (%)	BN	66.9 \pm 2.1	62.3 \pm 2.2	
	RNNSLU	74.6 \pm 1.9	63.1 \pm 2.2	
	RNNASR	79.6 \pm 1.8	71.8 \pm 2.0	
Number of turns	BN	6.02 \pm 0.15	6.61 \pm 0.14	
	RNNSLU	5.69 \pm 0.14	6.11 \pm 0.15	
	RNNASR	5.30 \pm 0.12	6.04 \pm 0.14	
Subjective reward	BN	11.09 \pm 0.38	9.91 \pm 0.41	
	RNNSLU	11.15 \pm 0.39	11.21 \pm 0.39	
	RNNASR	13.49 \pm 0.29	12.23 \pm 0.33	
Filtered reward	BN	10.76 \pm 0.44	8.87 \pm 0.49	
	RNNSLU	11.54 \pm 0.40	10.41 \pm 0.48	
	RNNASR	13.64 \pm 0.30	12.39 \pm 0.36	
“The system understood me well.” (0-4)	BN	2.99 \pm 0.05	2.82 \pm 0.06	
	RNNSLU	3.06 \pm 0.05	3.12 \pm 0.05	
	RNNASR	3.38 \pm 0.04	3.09 \pm 0.05	
Number of dialogues	BN	498	493	
	RNNSLU	512	485	
	RNNASR	496	485	

Table 6.8: Results for the 6 systems using policies learnt online. Errors are the standard error in the mean. A small visualisation of the results is given for each metric in the last column, where the systems are plotted from left to right – BN, RNNSLU, RNNASR, BN_degraded, RNNSLU_degraded, RNNASR_degraded. Top results are made bold in each group.

			BN	RNNSLU	RNNASR
Subjective success rate (%)			84.1 ± 1.2	85.4 ± 1.1	92.7 ± 0.8
<i>p</i> -values	BN		–	0.4	2×10^{-9}
	RNNSLU		–	–	2×10^{-7}
Objective success rate (%)			64.6 ± 1.5	69.0 ± 1.5	75.7 ± 1.4
<i>p</i> -values	BN		–	0.04	6×10^{-8}
	RNNSLU		–	–	8×10^{-4}
Number of turns			6.3 ± 0.1	5.9 ± 0.1	5.6 ± 0.1
<i>p</i> -values	BN		–	2×10^{-4}	1×10^{-8}
	RNNSLU		–	–	0.06
Subjective reward			10.5 ± 0.3	11.2 ± 0.3	12.9 ± 0.2
<i>p</i> -values	BN		–	9×10^{-4}	2×10^{-12}
	RNNSLU		–	–	5×10^{-4}
Filtered reward			9.8 ± 0.3	11.0 ± 0.3	13.0 ± 0.2
<i>p</i> -values	BN		–	1×10^{-4}	4×10^{-17}
	RNNSLU		–	–	8×10^{-6}
“The system understood me well.” (0-4)			2.90 ± 0.04	3.08 ± 0.04	3.24 ± 0.03
<i>p</i> -values	BN		–	2×10^{-4}	5×10^{-11}
	RNNSLU		–	–	5×10^{-3}

Table 6.9: Results for the 3 dialogue state trackers using policies learnt online, pooling across acoustic conditions. Results of Kruskal Wallis rank sum tests are given for each metric and all three pairs of values. A *p*-value (of rejecting the null-hypothesis that the values are indistinguishable) is made bold if it is lower than 3×10^{-3} . This threshold ($< 0.05/15$) is the typical 5% with Bonferroni correction as there are 15 comparisons in total.

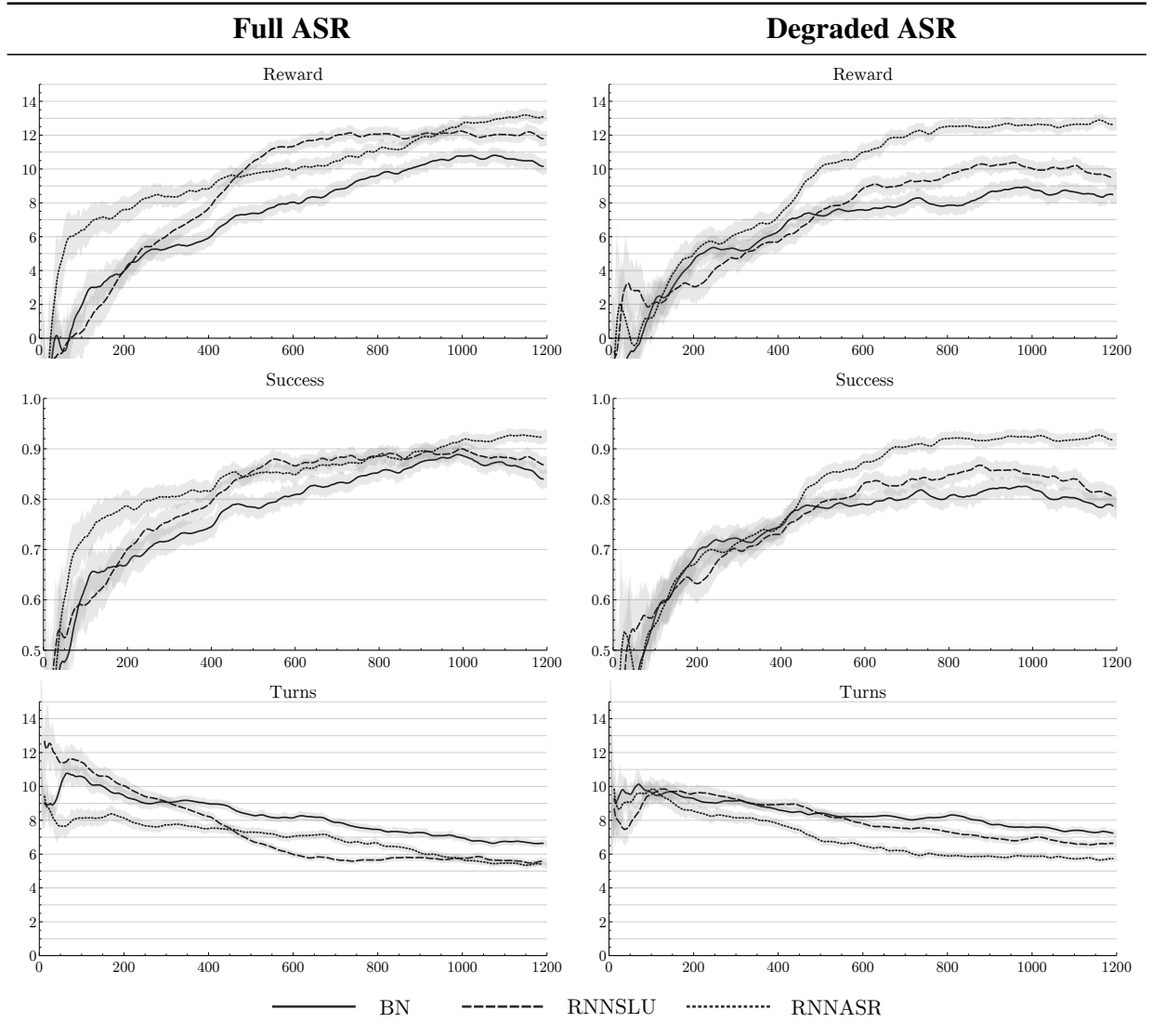


Figure 6.2: Moving average reward, success, and number of turns during the 1,200 learning dialogues for each of the 6 systems starting from a random policy. x -axis is number of learning dialogues, and grey area shows the standard error in the estimation of the moving average, estimated using 400 dialogues.

distributions (as measured by the L2 metric), possibly as a result of the joint optimisation, and by avoiding any information bottle neck arising from summarising the dialogue turn as a finite list of dialogue act hypotheses. Furthermore this method is readily applied to expanded domains by exploiting delexicalised inputs.

The promising results in the off-line evaluations were validated in a live user trial. A policy was learnt online for each DST configuration, optimising performance relative to the resulting dialogue state estimations. The use of discriminative DST is seen to give significant and substantial gains in end-to-end dialogue quality as measured by a variety of metrics.

CHAPTER 7

UNSUPERVISED ADAPTATION FOR RNN DIALOGUE STATE TRACKING

Though the discriminative Recurrent Neural Network (RNN)-based Dialogue State Tracking methods were shown in chapter 6 to outperform *Bayesian networks* in the off-line and live evaluations of the previous sections, there remains at least one advantage obtained by defining state tracking as a *generative* process. By running the expectation propagation algorithm over unlabelled dialogues, the parameters of a *dynamic Bayesian network* can be adapted in an off-line and unsupervised manner (Thomson et al., 2010a). Updating the parameters of an RNN on the other hand requires labelled training data, as the log-likelihood cost function used in Stochastic Gradient Descent (SGD) uses the identity of the true labels.

This chapter presents an approach to adapt the RNN parameters in a live dialogue system, without explicit labels, when deployed in a new domain. This is applied to *word-based* trackers that work directly on the words from the Automatic Speech Recognition (ASR), so that they might adapt to the language used to express constraints on the slots in the new domain.

Relevant fields in machine learning include semi-supervised learning (Zhu and Goldberg, 2009), which attempts to learn from unlabelled in-domain data, and transfer learning (Pan and Yang, 2010), which tries to exploit labelled out-of-domain examples. The techniques presented in those fields are typically designed for static classification tasks. The approach defined here however exploits specific patterns of dialogue sequences, and is formulated specifically for use in live Dialogue State Tracking.

7.1 Online Adaptation

In this context, models are initialised with a set of reasonable parameters W^{init} obtained by training on data for which labels are available. Adaptation is then the process of updating the parameters W^* having observed dialogues in the new domain. In online adaptation, W^* may be updated after each dialogue, but the parameters may not be updated using any data from any dialogue before the tracker has output its distributions for that dialogue.

Let \mathbf{f}_t denote the input features to the RNN at *turn* t , and define \mathbf{F} as the sequence of inputs for a dialogue: $\mathbf{F} = (\mathbf{f}_0, \dots, \mathbf{f}_{T-1})$ where T is the length of the dialogue. An RNN of a given structure can be considered as a function that takes a set of values for the weight and bias parameters, W , and maps \mathbf{F} to a sequence of distributions over the labels, $\mathbf{Y} = (\mathbf{y}_0, \dots, \mathbf{y}_{T-1})$. In particular, for the set of initial parameters W^{init} ,

$$\mathbf{Y}^{\text{init}} = \text{RNN}(W^{\text{init}}, \mathbf{F}) \quad (7.1)$$

Also define $\mathbf{Y}^* = (\mathbf{y}_0^*, \dots, \mathbf{y}_{T-1}^*)$ by the following relation:

$$\mathbf{Y}^* = \text{RNN}(W^*, \mathbf{F}) \quad (7.2)$$

Unsupervised adaptation is facilitated by defining a scoring criterion that evaluates the adapted parameters W^* without requiring any labels.

7.1.1 Criterion for Unsupervised Adaptation

The proposed criterion $C(W^*)$ used to score a set of parameters W^* (without requiring labels) is defined as follows:

$$C(W^*) = \left(\sum_{t=0}^{T-1} H(\mathbf{y}_t^{\text{init}}) H(\mathbf{y}_t^*, \mathbf{y}_{t+1}^{\text{init}}) \right) + \lambda \|W^* - W^{\text{init}}\| \quad (7.3)$$

where $H(\mathbf{y}) = -\sum_i y_i \log y_i$ is the entropy of the distribution \mathbf{y} and $H(\mathbf{y}, \mathbf{y}') = -\sum_i y_i \log(y'_i)$ is the cross-entropy between \mathbf{y} and \mathbf{y}' . $\|W\|$ is the norm of the parameters W , e.g. the l_2 or l_1 norm.

The regularisation term multiplying $\lambda > 0$ enforces that W^* should stay close to W^{init} . In the sum, the only terms that change with W^* are the cross-entropies $H(\mathbf{y}_t^*, \mathbf{y}_{t+1}^{\text{init}})$. As \mathbf{y}^{init} is fixed, the cost is minimised if $\mathbf{y}_{t+1}^{\text{init}} = \mathbf{y}_t^*$ at each t . Therefore in minimising C the parameters W^* receive a learning signal to adapt \mathbf{y}_t^* towards $\mathbf{y}_{t+1}^{\text{init}}$, weighted by $H(\mathbf{y}_t^{\text{init}})$.

At turns where the initial model is uncertain (high entropy), minimising C will attempt

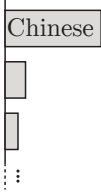
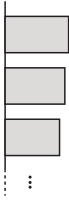
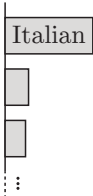
Dialogue Turn	\mathbf{y}^{init}	Notes
Turn 1 <i>System:</i> What type of food would you like? <i>User:</i> Chinese food.		Here an initial model is likely to output a confident low entropy distribution correctly identifying the food goal as <i>Chinese</i> .
Turn 2 <i>System:</i> There are no matching Chinese restaurants. <i>User:</i> Any serving pizza?		The system has requested the food slot, and the user's response included the term 'serving'. This gives evidence that the user has informed the food slot, but the system does not understand that 'pizza' implies Italian food. Therefore it is likely that an initial model would output a high entropy distribution for the food slot.
Turn 3 <i>System:</i> Sorry, what type of food would you like? <i>User:</i> Um, Italian food.		If the user explicitly says 'Italian', which the system is able to match in the domain, then an initial model can predict with high confidence the correct value for food is <i>Italian</i> .

Figure 7.1: Example dialogue where user guidance can be exploited in unsupervised learning. The entropy of the food slot is high in the second turn, and low in the third peaking at 'Italian'. Therefore minimising C (equation (7.3)) leads to a learning signal that correlates the n -grams in the second turn ('Something serving pizza') with the 'Italian' hypothesis for food. The low entropy of the slot in the first turn causes the corresponding term in C to diminish, so the signal does not lead to a correlation with the n -grams in the first turn.

to find parameters that can predict the next turn's label according to the output of the initial model. To illustrate this, figure 7.1 gives an example dialogue sequence where this leads to improved parameters, W^* . In general the minimisation of C works to propagate a learning signal from later turns in the dialogue, where an initial model should be more sure about the user's goal, to earlier turns. The term $H(\mathbf{y}_{t+1}^{\text{init}})$ ensures that this signal should stop at turns where the initial model is confident about its prediction.

7.1.2 Schedule for Online Adaptation

The following procedure performs online adaptation of the parameters while classifying a set of dialogues. This is used for tracking dialogues in the Dialog State Tracking Challenge (DSTC) data, and could also be used in a live system. It works by collecting dialogues in batches of size N , and updating the parameters using these batches.

1. Set $W^* = W^{\text{init}}$ and $D = \emptyset$.
2. Track the next dialogue using parameters W^* . Append the dialogue (represented by the input feature sequence \mathbf{F}) to the batch, $D \rightarrow D \cup \{\mathbf{F}\}$. If $|D| = N$, then enter step 3, otherwise return to step 2.
3. Update W^* using SGD to minimise $C(W^*)$ over the dialogues in D . Reset D to \emptyset , and return to step 2.

Note that this algorithm uses a fixed W^{init} that is not updated. It is possible to convert this into an incremental algorithm, by setting W^{init} to W^* after each batch, or after every n batches. Empirically, it seems that the parameters diverge too far if W^{init} is updated after every batch. Preliminary testing suggests that reasonable results are obtained if W_{init} is updated after a larger number of batches. However, for simplicity, W_{init} is considered fixed in the work reported here, and an analysis of incremental adaptation is left for future work.

7.2 Evaluation

This section evaluates the proposed method for online and unsupervised adaptation to expanded domains in Dialogue State Tracking. The method is initially validated using data from DSTC 2, emulating an expanded domain by removing the labels for the *food* slot during training. Results on testing in an expanded domain are then presented on the DSTC 3 data.

7.2.1 Adaptation with Missing Labels

Here adaptation is evaluated using data from a single domain. A set of initial trackers is trained without the labels for the *goal constraint* of slot s , which are then adapted to the task of tracking the slot s . Though this is a fairly artificial setup, it may give some confidence that adaptation will be beneficial during actual deployment in an expanded domain (see next section). It is also conceivable that only partially labelled data may be available in some cases during training due to e.g. the prohibitive cost of obtaining labels for slot s .

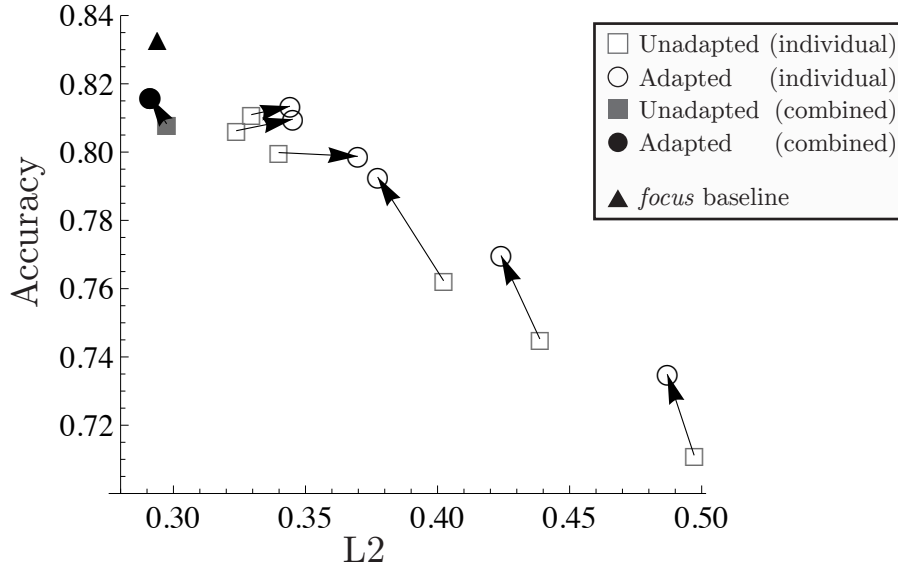


Figure 7.2: Accuracy and L2 scores on the *food* slot for trackers trained using only labels for *area* and *pricerange*. Note lower L2 scores are better. The results for individual RNNs are shown as outlined shapes. Squares show the performance for initial RNN trackers (*Unadapted*) connected to circles showing the performance using unsupervised adaptation (*Adapted*). The performance for combining the groups using score averaging are also shown. The *focus* baseline was the strongest baseline in DSTC 2. This baseline assumes a semantic decoder for the *food* slot, which the other trackers are not given.

This study investigates training *delexicalised* word-based RNNs for the *food* slot using only labels for *area* and *pricerange* in the DSTC 2 training sets. Though the RNNs are never exposed to labels for the *food* slot, they are used to classify the *food* goal constraint in the DSTC 2 test set. The *food* slot is chosen as it is an outlier in the domain; while *area* and *pricerange* have cardinalities 5 and 3 respectively, there are 91 possible food types. This contributes to the food slot being the hardest to track (in the DSTC 2 evaluation, trackers consistently scored lowest on food of all slots), and the most challenging to label.

An ensemble of six delexicalised RNNs, with varying parameters as described in section 6.3, were trained using ASR features and the labels for *area* and *pricerange*. Each of these can immediately be used to track the *food* slot without any unsupervised adaptation, which is called the *Unadapted* condition. Each of the six RNNs can also be adapted online during the classification of the test set using the procedure described in section 7.1, which is called the *Adapted* condition.

Figure 7.2 illustrates the performance of the six trackers with and without online unsupervised adaptation at test time. In all cases, adaptation improved accuracy on the test set. On average the accuracy improved by 1.4% and in the best case by 3.0%. The L2 score,

		New slots	Old slots	Joint
<i>mean</i>				
	Unadapted	0.866	0.877	0.552
	Adapted	0.869	0.890	0.566
<i>combined</i>				
	Unadapted	0.893	0.900	0.616
	Adapted	0.892	0.900	0.623

Table 7.1: Goal constraint tracking accuracies on the DSTC 3 test set for word-based RNN trackers, with and without unsupervised adaptation. Accuracy is reported on the *Old slots* and *New slots*, i.e. slots found and not found in the training set respectively. The joint goal constraint accuracy is also given. In each condition, a group of six RNNs are trained. Mean accuracies for the group are reported, as well as the accuracies of the groups combined using model averaging. For *mean* results, bold denotes a difference of over 2 standard errors.

which measures the quality of the probability scores, was on average improved but in certain individual cases the L2 score deteriorated slightly.

The two groups of six were combined using score averaging. The combined results also demonstrate a slight improvement using adaptation, with the combined *Adapted* group giving the best tracking performance. The performance is comparable with many of the entries in DSTC 2 including the *focus* baseline (the strongest baseline), and is achieved without any training labels for the slot.

7.2.2 Adaptation to New Domains

This section shows results in applying unsupervised adaptation to the delexicalised word-based RNN tracker described in section 6.3. This tracker is trained on the DSTC 2 data in the restaurant information domain, then evaluated on the DSTC 3 test set in the tourist information domain. Recall the tourist information domain is considered an expanded domain, which subsumes restaurant information. It introduces two new types of venue (hotels and coffee shops) and contains 5 additional slots.

Table 7.1 summarises the results on the DSTC 3 test set. Unsupervised adaptation is shown to give an improvement in the mean accuracy for the old slots (those in the training data) and the joint. After combination using score averaging, the top joint goal constraint accuracy of 0.623 among the word-based ASR trackers is obtained by the combined adapted tracker.

No gain from adaptation is found on the new slots for the combined trackers. Table 7.2 gives the performance on each slot for the combined trackers. These results suggest that the

Slot	Unadapted	Adapted
<i>Old slots</i>		
area	0.845	0.848
food	0.896	0.897
name	0.890	0.890
pricerange	0.925	0.921
<i>New slots</i>		
children allowed	0.644	0.660
has internet	0.793	0.818
has tv	0.788	0.762
near	0.834	0.801
type	0.935	0.936

Table 7.2: Performance of the combined unadapted and adapted trackers of table 7.1, in terms of goal constraint accuracy for each slot in the domain.

adapted combined tracker performs similarly to the unadapted tracker on all of the *old slots*. Adaptation is able to improve the accuracy in some cases for *new slots* such as *children allowed* and *has internet*, while compromising performance on other slots such as *has tv* and *near*.

Note that on average the individual trackers give improved performance on the old slots (the *mean* results of table 7.1), but this improvement does not carry over to the results after model averaging (the *combined* results).

Though the overall joint goal constraint accuracy is improved, the resulting L2 score is 0.586. Comparing this to table 6.5, it seems there is some trade-off between accuracy and the quality of the scores as probabilities, as measured by the L2 score, when using the unsupervised adaptation.

7.3 Conclusions

This chapter has proposed a criterion for unsupervised adaptation that, when optimised with SGD, serves to propagate information from later on in a dialogue to earlier turns. This gives a technique for improving the parameters of the RNN without requiring training labels.

Though this technique for online unsupervised adaptation gives somewhat promising results in the evaluation, the improvements in tracking accuracy are small. Other forms for the unsupervised training criterion can also be imagined. For example an explicit model for how likely a dialogue state is to persist from turn to turn could be used, rather than the cross-entropy term in equation (7.3) (perhaps similar to the *durability* term in Kadlec et al.

(2014b)).

Partial labels may also be useful for adaptation. For example, if the dialogue state is known in a subset of turns in a dialogue, conventional SGD on the log probability of the whole sequence can be used to improve the parameters of an RNN. This suggests an active learning approach (Settles, 2009; Tur et al., 2003), where individual dialogue turns are selected for labelling. As dialogue state labelling is an expensive task, learning from carefully selected dialogue turns may be worthwhile.

CHAPTER 8

STRUCTURED OUTPUTS FOR RNN DIALOGUE STATE TRACKING

In the proposed Recurrent Neural Network (RNN)-based framework for dialogue state tracking, an independent RNN is trained for each individual slot. To obtain the joint distribution over *goal constraints*, the outputs from each RNN are simply multiplied. In this way, the dialogue state is approximated as a product of marginal distributions over sub-components of the dialogue state.

It may not always suffice to make this approximation; certain domains may benefit from a full structured joint distribution over the sub-components of the dialogue state. For example, there may be two mutually exclusive goal constraints $s_0 = v_0$ and $s_1 = v_1$. A structured joint prediction could specify that *either* the user wants $s_0 = v_0$ *or* $s_1 = v_1$, while this cannot be expressed as a product of two marginal distributions over s_0 and s_1 . In the tourist information domain (see appendix A), it only makes sense to specify the food type if the correct venue type is *restaurant*, which is an example of this kind of dependency between slots.

In general, learning a joint model may allow for better modelling of the correlation between slots. As well as hard constraints, as mentioned above, this might improve tracker accuracy by exploiting observations such as users looking for expensive food in Cambridge are perhaps more likely to ask for French cuisine than Chinese.

This chapter introduces a method to allow the RNN-based dialogue state trackers to output a single full structured joint distribution over multiple sub-components of the dialogue state. Section 8.1 presents how this has been achieved in other frameworks for dialogue state tracking. Section 8.2 presents the additional neural network structure used to join multiple independent RNNs to output a single structured joint distribution. An evaluation of the proposed method is given in 8.3 using data from the Dialog State Tracking Challenge (DSTC) 2 evaluation.

8.1 Related work

Dependencies between slots can be easily incorporated in generative models, such as the Bayesian Update of Dialogue State (BUDS) system described in section 5.3, by connecting slots in the *Bayesian network*. For the tourist information domain, certain slots are only applicable for venues of certain *types*. For example *food* is only applicable for restaurants, and *has internet* is only applicable for pubs and coffee shops. The Bayesian network is configured so that the goal constraint for the *type* slot is a parent for all other slots. A simple hand-crafted distribution is then specified, for example of the form:

$$P(g_s = v_s | g_{\text{type}} = v_{\text{type}}) \propto \begin{cases} 1 - \theta & \text{if } s \text{ is an applicable slot for } v_{\text{type}} \\ 1 - \theta & \text{if } s \text{ is not an applicable slot for } v_{\text{type}} \text{ but } v_s = \text{None} \\ \theta & \text{otherwise} \end{cases} \quad (8.1)$$

Other dependencies between slots could also be included, though the parameters of the new conditional probability distributions would have to be either learnt or specified by hand.

The ranking approach for Dialogue State Tracking (DST) presented in Williams (2014) is one of the few entries in the DSTCs to use a structured joint distribution for the goal constraints, rather than calculating the joint distribution as a product of independent marginals (Henderson et al., 2014b). In this approach, a set S of possible joint hypotheses for the goal constraints is generated using the hypotheses suggested by the Spoken Language Understanding (SLU). An element of S would be for example $\{food=Chinese, area=west\}$. Features are extracted for each of these hypotheses, and each is scored with a *discriminative* ranking classifier. More details of the feature extraction is given in section 5.4. The output is then a distribution over structured joint predictions for the goal constraints (the elements of S). Williams finds gains in *accuracy* from using structured output over independent outputs, though at the expense of worse *L2* scores.

8.2 Structured Joint Predictions

Consider a set of discrete random variables $X = \{x_0, \dots, x_{n-1}\}$, each with a finite set of possible values. Each random variable x_i has possible values $\{0, 1, \dots, |x_i| - 1\}$. The joint

distribution can be written as:

$$P(x_0 = x'_0, \dots, x_{n-1} = x'_{n-1}) = P(x_0 = x'_0) \prod_{i=1}^{n-1} P(x_i = x'_i | x_0 = x'_0, \dots, x_{i-1} = x'_{i-1}) \quad (8.2)$$

which is just one expansion of the full joint distribution. Note that the conditional probabilities in the product require learning $\prod_{j=0}^i |x_j|$ individual probabilities. This does not scale well with the number of variables and their dimensions. If each random variable takes m values, this gives $O(m^n)$ parameters to learn.

Bengio and Bengio (2000) present a method of modelling such structured joint probability distributions, which scales more favourably. This method uses a neural network structure to model the joint distribution in analogy to equation (8.2) as follows:

$$P(x_0 = x'_0, \dots, x_{n-1} = x'_{n-1}) = p_{x'_0} \prod_{i=1}^{n-1} \text{softmax}(\text{NNet}(\mathbf{z}_0 \oplus \dots \oplus \mathbf{z}_{i-1}))_{x'_i} \quad (8.3)$$

where \mathbf{p} is a learnt vector of size $|x_0|$, and \mathbf{z}_i is a vector encoding of the assignment $x_i = x'_i$. The notation follows appendix D. In this work, \mathbf{z}_i is a $|x_i|$ dimensional *1-hot* encoding:

$$z_{i,j} = \begin{cases} 1 & \text{if } x'_i = j \\ 0 & \text{otherwise} \end{cases} \quad (8.4)$$

The sub-networks $\text{NNet}(\mathbf{z}_0 \oplus \dots \oplus \mathbf{z}_{i-1})$ output a vector of size $|x_i|$, which is then input to a softmax function to give a probability distribution over the $|x_i|$ possible values for x_i . Each of these sub-networks (indexed by i) has an input of size $\sum_{j=0}^{i-1} |x_j| \sim O(im)$, and an output of size m . If each has one hidden layer of fixed size h , then the number of parameters for the neural network is $O(imh + hm)$, so the total number of parameters is $O(hmn^2)$.

This model is therefore an efficient way of modelling joint distributions, which Bengio and Bengio found to be effective for several datasets. Note that the definition of the model relies on an ordering of the random variables. In this work, random orderings are used, and the ordering is not found to affect results.

Equation (8.3) provides a method of learning joint distributions using neural networks, but DST requires learning a joint distribution *conditioned* on the evidence of the dialogue so far. This requires an adaptation of this model, given in the next section.

8.2.1 Conditional Structured Joint Predictions

For tracking goal constraints, the random variables $x_{0,t}, \dots, x_{n-1,t}$ are the informable slots in the domain at each turn t . In the notation of section 5.3, these are the variables $g_{s_0,t}, \dots, g_{s_{n-1},t}$. Assume an RNN structure for each slot. So for a dialogue consisting of T turns, the RNNs give outputs $\mathbf{P}_i^{\text{indep}} = (\mathbf{p}_{i,0}^{\text{indep}}, \dots, \mathbf{p}_{i,T-1}^{\text{indep}})$ for each slot x_i . The joint distribution for turn t is then modelled as:

$$\begin{aligned} P(x_{0,t} = x'_{0,t}, \dots, x_{n-1,t} = x'_{n-1,t} \mid \text{dialogue up to turn } t) \\ = p_{0,t,x'_{0,t}}^{\text{indep}} \prod_{i=1}^{n-1} \text{softmax} \left(\log(\mathbf{p}_{i,t}^{\text{indep}}) + \text{NNet}(\mathbf{z}_0 \oplus \dots \oplus \mathbf{z}_{i-1}) \right)_{x'_i} \end{aligned} \quad (8.5)$$

where the logarithm acts component-wise, and \mathbf{z}_i is again a vector encoding of the assignment $x_{i,t} = x'_{i,t}$. This is an adaptation of equation (8.3) to include the output of the RNNs at turn t . Note that if the sub-networks $\text{NNet}(\mathbf{z}_0 \oplus \dots \oplus \mathbf{z}_{i-1})$ output $\mathbf{0}$, then this equation reduces to the independent case:

$$\begin{aligned} P(x_{0,t} = x'_{0,t}, \dots, x_{n-1,t} = x'_{n-1,t} \mid \text{dialogue up to turn } t) \\ = p_{0,t,x'_{0,t}}^{\text{indep}} \prod_{i=1}^{n-1} \text{softmax} \left(\log(\mathbf{p}_{i,t}^{\text{indep}}) \right)_{x'_i} \\ = \prod_{i=0}^{n-1} p_{i,t,x'_i}^{\text{indep}} \end{aligned} \quad (8.6)$$

and so regularisation of the parameters of the network, which drives the output of the $\text{NNet}(\mathbf{z}_0 \oplus \dots \oplus \mathbf{z}_{i-1})$ sub-networks to $\mathbf{0}$, encodes a preference for defaulting to the independent case, where the joint is calculated as a product of marginals. The $\text{NNet}(\mathbf{z}_0 \oplus \dots \oplus \mathbf{z}_{i-1})$ sub-networks modify the approximation to allow modelling a structured joint distribution.

This method can be considered as a joint prediction output network, which connects the outputs of the independent individual RNNs at each turn.

The log probability of the correct joint hypothesis as given by equation (8.5) gives a new objective to maximise in the Stochastic Gradient Descent (SGD) algorithm. This objective can be used not only to optimise the parameters of the sub-networks $\text{NNet}(\mathbf{z}_0 \oplus \dots \oplus \mathbf{z}_{i-1})$, but also the RNNs for each slot. Therefore after training with this criterion, the $(\mathbf{p}_{i,t}^{\text{indep}})$ predictions may change.

In training, the parameters of the RNNs for each slot are first optimised using the independent log probabilities (equation (8.6)) until the parameters have converged. The structured joint probability (equation (8.5)) is then used for optimisation until convergence, learning the new parameters and adapting the RNN parameters jointly. It would also be possible to optimise equation (8.5) directly from a random initialisation, however the outlined scheme is found to be more efficient in terms of computational time required for convergence.

For tracking, this chapter only considers outputting a single best hypothesis. A beam search is performed to find the assignment that maximises equation (8.5), by selecting assignments to each slot in turn. The beam search operates as follows:

1. Set $P = \emptyset$, the *beam* of partial hypotheses for the joint assignment to X . A *beam width* N is chosen, the maximum size of P at each step. A partial hypothesis in P is a tuple (J, p) , where J is a sequence of integers representing an assignment to the first $|J|$ variables in X ($x_i = J_i$), and p is the probability score computed from equation (8.5). The notation $J \oplus (j)$ denotes the sequence obtained from appending the integer j to the sequence J .
2. Calculate $\mathbf{p}_{0,t}^{\text{indep}}$. Write j_0, \dots, j_{N-1} for the indices of the top N elements of $\mathbf{p}_{0,t}^{\text{indep}}$. Let $P = \{((j_k), p_{0,t,j_k}) \mid k = 0, \dots, N-1\}$.
3. Let $i = 1$.
4. Calculate $\mathbf{p}_{i,t}^{\text{indep}}$. Let $P' = \emptyset$.
5. For each partial hypothesis $(J, p) \in P$:
 - (a) Encode the assignment in J as vectors $\mathbf{z}_0, \dots, \mathbf{z}_{i-1}$, i.e. $z_{j_k} = 1$ if $k = J_j$ or 0 otherwise for $k = 0, \dots, (|x_j| - 1)$.
 - (b) Calculate $\mathbf{p} = \text{softmax} \left(\log(\mathbf{p}_{i,t}^{\text{indep}}) + \text{NNet}(\mathbf{z}_0 \oplus \dots \oplus \mathbf{z}_{i-1}) \right)$. Write j_0, \dots, j_{N-1} for the indices of the top N elements of \mathbf{p} . Add to P' the corresponding new partial hypotheses, $P' \leftarrow P' \cup \{(J \oplus (j_k), p \times p_{j_k}) \mid k = 0, \dots, N-1\}$.
6. Set P to the top N partial hypotheses in P' , i.e. the $(J, p) \in P'$ with the highest values for p .
7. Increment i by 1 ($i \leftarrow i + 1$). If $i < n$, then return to step 4.
8. Select the top partial hypothesis in P as the chosen assignment for X , i.e. $(J, p) \in P$ with maximum p .

In this work, a beam search with $N = 1$ is performed, as this is found to identify the top scoring hypothesis in the vast majority of cases.

8.3 Evaluation

This section presents an evaluation of the proposed approach for joining multiple independent RNN classifiers to output a structured joint distribution at each turn. The DSTC 3 data is split into a training set of 2,049 dialogues and an evaluation set of 226 dialogues.

The DSTC 3 data is chosen because structured joint predictions are more likely to be important in the tourist information domain than in the restaurant information domain of DSTC 2. For example, specifying a food type should only make sense if *type = restaurant*, and people should not ask for *has internet = true* if they also require *type = restaurant*. Such rules do not exist in the restaurant information domain.

On the evaluation set, the focus baseline (see section 5.2.3) achieves a joint goal constraint accuracy of 0.558, while the Bayesian network obtains an accuracy of 0.579.

Table 8.1 presents the results for both *word-based* and conventional SLU input RNN-based tracking. An ensemble of six RNNs were trained for each condition, with varying parameters as described in section 6.3. A single hidden layer is used for the sub-networks in equation (8.5) of size 64. A modest improvement is seen in all cases from using a structured model for the joint distribution. This is more pronounced in the case of using SLU input features.

	Mean	Combined
<i>SLU features</i>		
Independent	0.585	0.607
Structured	0.599	0.622
<i>ASR features</i>		
Independent	0.679	0.711
Structured	0.685	0.717

Table 8.1: The joint goal constraint accuracy on the 226 evaluation dialogues achieved by independent and structured RNN models. In total 6 trackers are trained for each input type – SLU features and ASR features (word-based tracking). Each of the 6 trackers uses a randomly selected ordering of the slots. The mean metrics among the 6 trackers are shown as well as the results from combining the 6 trackers using majority voting.

In the majority of cases where the structured joint predictions improve performance, it is found that the structured output differs from the independent output in that it correctly

identifies the *None* hypothesis for a slot. Recall the *None* hypothesis means that the user is yet to specify a constraint for the slot.

Slot	None accuracy	
	Independent	Structured
area	0.913	0.975
children allowed	0.988	0.996
has internet	0.988	0.993
has tv	0.996	0.999
food	0.963	0.991
name	1.000	1.000
near	0.982	0.996
type	0.800	0.800

Table 8.2: None accuracy of the output of SLU-based RNNs for both independent and structured joint outputs.

Slot	None accuracy	
	Independent	Structured
area	0.970	0.985
children allowed	0.989	0.999
has internet	0.992	0.992
has tv	0.948	0.991
food	0.947	0.986
name	0.999	1.000
near	0.984	0.992
type	0.823	0.823

Table 8.3: None accuracy of the output of word-based RNNs for both independent and structured joint outputs.

Tables 8.2 and 8.3 investigate the *None accuracy* for the RNN trackers with independent and structured outputs. The None accuracy is the fraction of turns where a tracker correctly identifies the *None* hypothesis for the given slot. In all cases the structured output improves this metric, in particular for the *area*, *near*, and *food* slots.

The improvement for *area* and *near* slots may be due to the inherent correlation between the two slots. It only makes sense to ask for a venue in a certain *area* and located *near* another point if that point is in the same *area*. Therefore the structured model may learn to change the less reliable hypothesis to *None* if the two are conflicting. The improvement for the *food* slot may be principally explained by its correlation with the other slots. For

example certain cuisines are only available in certain areas, and only for venues with *type = restaurant*.

8.4 Conclusions

This chapter has presented a method of extending classifiers trained to classify individual random variables to a single joint model outputting a structured joint distribution over the variables. This is applied to the RNNs used for DST, though it could be useful in other classification tasks.

The technique gave small improvements in the quality of the top hypothesis for the user’s goal constraints in dialogue data in the restaurant information domain. In particular, the structured model improved on the independent model by better learning when goal constraint hypotheses should be changed to *None* for particular slots. The benefit from learning a joint model may be more substantial in more complex domains where slot correlations are more important.

One key disadvantage of this method is that it requires all the slots to be present in the training data when optimising the joint probability. It is therefore more challenging to extend this model to expanding domains than it is for individual independent classifiers.

Bengio and Bengio found improvements in excluding some \mathbf{z}_j from the input to the sub-networks of equation (8.3). This effectively excludes dependencies between pairs of variables, and the exclusions can be selected using standard statistical analyses or by the system designer. It is possible that such pruned neural networks might be useful for equation (8.5).

It was found that using a structured output improved the model’s ability to correctly identify the *None* hypotheses, i.e. which slots have not yet been specified by the user. This suggests that limiting the output of the sub-networks of equation (8.5) to only adjust the *None* hypotheses may result in a network with far fewer parameters and comparable performance.

This chapter has looked at outputting only a single hypothesis for the joint goal constraints. However, an N -best list of DST hypotheses could be generated using equation (8.5). Many implementations of Partially Observable Markov Decision Process (POMDP) for spoken dialogue systems summarise the dialogue state as the marginal distributions over each goal constraint (Gašić et al., 2010; Thomson, 2009). In this case the N -best list would then have to be marginalised for each slot. Though this would inherently lose some information, it is still possible that the marginalised distributions would be more accurate than if they were the direct output of classifiers as in independent models.

It may be advantageous to adapt equation (8.5) so that the sub-networks in the product also take the independent marginals $\mathbf{p}_{t,i}^{\text{indep}}$ as input. This would allow for more complex interactions with the marginal distributions when modelling the structured joint distribution.

CHAPTER 9

CONCLUSIONS

This thesis has shown that *discriminative* methods can significantly improve the performance of a statistical dialogue system. Unlike traditional *generative* models for the tasks of Spoken Language Understanding (SLU) and Dialogue State Tracking (DST), discriminative models are able to incorporate arbitrary potentially useful features. They directly optimise the quantities that are important in statistical dialogue systems, the conditional probabilities over the outputs of a component given the output of the previous component in the pipeline.

The original contributions of this thesis include the *CNet decoder* for SLU operating directly on the *word confusion network*, and applying Recurrent Neural Networks (RNNs) models to DST, including *word-based* tracking, unsupervised adaptation, and methods for learning structured outputs.

The proposed CNet decoder enables improved performance in SLU. Discriminative models allow for the use of arbitrary features, and top performance is achieved by using features derived from the full posterior distribution of the Automatic Speech Recognition (ASR), as well as dialogue context features. Both off-line evaluations and live user trial show the discriminative method for SLU outperforms conventional methods.

For DST, the word-based RNN tracker is shown to perform well relative to approaches proposed by other research teams in the Dialog State Tracking Challenges (DSTCs). Discriminative DST using RNNs is shown in a live evaluation to give significant and substantial improvements in end-to-end dialogue quality. The word-based RNN tracker fuses SLU and DST into one discriminative model that can be learnt jointly. In mapping directly from words to the dialogue state, it has the advantage of not requiring any intermediate semantic representations. This removes the need to define taxonomies of dialogue acts. Experiments show that the RNNs can be readily applied to new dialogue domains, by using special *delexicalised* features, which generalise across slots and values. An algorithm is presented that shows how the RNN parameters can be adapted during interactions without any labels, so

that unlabelled dialogue data can still be exploited. It is also shown how these networks can be adapted to output structured joint distributions over possible dialogue states.

9.1 Current Limitations and Future Work

One of the major issues with applying discriminative methods to the tasks of SLU and DST is obtaining the labelled data required to train the models. This has amounted to labelling utterances with true dialogue acts. This process can be facilitated by using crowd-sourced transcriptions and using accurate hand-crafted grammars on the first pass (see section 3.2.1). However, in the author’s experience, hand labelling hundreds of utterances with dialogue acts is required. As it stands, this job requires an expert in the field, and is not possible to crowd-source completely. Another issue with this technique for labelling is that it still relies on dialogue acts, while word-based tracking promises to eradicate the need for such intermediate semantic forms. Improvements to labelling may include formulating the task in a manner suitable for crowd-sourcing. Future work may also include cleverly selecting turns in the dialogue data for which labels would be most useful using techniques from *Active Learning* (Settles, 2009).

Some work has been presented in applying and adapting trackers to expanded domains, i.e. domains that subsume the domain used for training but include multiple new slots. In this case, the expanded domain is assumed to be fully specified. It would be interesting to investigate methods for learning from data which expansions to an existing domain could be beneficial, in work similar to Chen et al. (2014), before learning how to track the expanded dialogue state in the learnt expanded domain.

This thesis has used a very simple method of deriving delexicalised features for word-based DST, performing exact string matching against a set of *aliases* (see appendix C). This method has two key deficiencies. The first is that the aliases are defined by the system designer. Future work may investigate automatically learning sets of possible expressions for slots and their values. The second issue is that exact string matching is potentially a brittle method to employ. There may be benefit in using soft string comparisons, which compare the similarity of phrases using lexical, syntactic, semantic, and phonetic distance measures.

The CNet decoder and word-based RNN tracker presented here have relied heavily on n -gram features, phrases consisting of 1, 2, or 3 words. These have provided an adequate representation for discerning the meaning of user utterances in the tourist information systems used for evaluation. However, n -gram representations may not be sufficient in more complex domains, where syntax is more important and users must express their goals with more complex language. In particular, the feature representations used provide limited information about the ordering of the words in the input sentence. Models that internally use a parse tree to understand language (see section 2.2) are psycholinguistically well-motivated

and promise to model many complex linguistic phenomena (Zettlemoyer and Collins, 2007).

For more complex domains, it may be necessary to use a model that deals with the sentence as a sequence. This could be achieved using a similar RNN-based model to that presented in section 5.5, whose recurrent time steps are at each word in the input (or perhaps each node in a word confusion network), as well as at each dialogue *turn*. A variety of RNN architectures could be employed, including *bidirectional* RNNs that can incorporate information from earlier and later words in the input sentence (Mesnil et al., 2013). In order to simulate the process of grammatically parsing the input sentence in the hidden structure of the RNN, it may be necessary to employ more complex models of memory such as deep long short-term memory gates (Hochreiter and Schmidhuber, 1997). Recent work has defined RNN models that can read and write from structured external memory resources (Graves et al., 2014). These models are able to learn simple algorithms such as copying, sorting, and associative recall using Stochastic Gradient Descent (SGD) and datasets of examples. A similar model could be imagined that defines differentiable versions of the data structures and operations required to parse a sentence using a grammar, with the advantage that this could be a sub-network of a larger RNN directly trained to track the dialogue state.

Alternatively, the idea of parsing could be exploited without adapting the model, but by designing feature representations. This is an advantage of discriminative methods, which allow the system designer to focus on improved feature representations rather than having to design more complex models (and derive new learning algorithms). Future work might look at improved representations for more complex tasks, e.g. exploiting parse trees (Kambhatla, 2004) and external resources such as paraphrase tables (Ganitkevitch et al., 2013).

This thesis has focussed on slot-based dialogue systems, where the dialogue can be modelled almost entirely by specifying constraints on slots and values (see appendix A). This formulation may not be possible for more open conversational dialogue systems. However, DST using RNNs could be viewed as a method of robustly tracking arbitrary random variables of interest in decision making (such as the *search method*), throughout the course of a dialogue sequence. Conversational systems might be able to exploit a set of accurately tracked and useful variables, such as the general topic of conversation. It seems likely that an altogether different approach than the slot-based framework may be needed to allow for open conversational systems that are not task-oriented and that can talk about anything.

Another limitation of the dialogue systems studied in this thesis is that they model dialogue as a strict turn-by-turn process between two participants (the user and the system). The boundary between turns in these systems is decided by a Voice Activity Detection classifier, which decides when the user has started and finished speaking. Multi-party dialogues are of interest, as are systems which have a more flexible notion of turn taking. Systems

may benefit from being able to interrupt the user or by giving back-channels while the user is talking. An RNN may still be an appropriate model to deal with such dialogues, where the time steps are more frequent than whole turns, and potentially labelled with the speaker's identity. Future work may involve applying RNN tracking for such dialogues, and might require reinforcement learning to allow the model to make turn taking decisions.

Despite current limitations, the proposed word-based RNN tracker may be particularly useful for system designers looking to apply their systems to new or expanding domains. Shared learning across slots and values, facilitated by the delexicalised features, gives a method for creating an initial dialogue system in a new domain using out of domain training data. As dialogues are collected in the new domain, these models can be rapidly improved to give state-of-the art performance. It is the hope that easily extensible models like this will allow for spoken interfaces whose capabilities will naturally grow as they are exposed to more interactions and knowledge of the world.

APPENDIX A

SLOT-BASED DIALOGUE DOMAINS

The slots of a slot-based dialogue system specify its domain, i.e. the scope of what it can talk about and the tasks that it can help the user achieve. The slots inform 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 5.1).

The following definitions assume the domain of the dialogue system is to allow the user to search a database of entities by specifying constraints. In this case the slots correspond to attributes of entities in the database.

The set of all slots S , is composed of two not-necessarily disjoint subsets – the requestable slots S_{req} , and the informable slots S_{inf} , such that $S = S_{req} \cup S_{inf}$. Informable slots are attributes of the entities in the database that the user may use to constrain their search. 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 ("*I'm looking for somewhere with the phone number 01223 715 715*"). For each slot $s \in S$, the set of possible values for the slot is denoted V_s .

The two domains used for evaluations in this thesis are the *restaurant information* and *tourist information* domains. The restaurant information domain was the domain used in the second Dialog State Tracking Challenge (Henderson et al., 2014b) and concerns finding restaurants in Cambridge, and the tourist information domain is the domain of the third Dialog State Tracking Challenge (Henderson et al., 2014d), which includes restaurants, pubs and coffee shops. The slots in tourist information are a superset of those in restaurant information, though the possible values V_s are different between the two domains even for slots present in both. A summary of the two domains is given in table A.1.

	Slot	$ V_s $	
		Restaurant	Tourist
S_{inf}	type	1	3
	area	5	15
	food	91	28
	name	113	163
	pricerange	3	4
	children allowed	-	2
	has internet	-	2
	has tv	-	2
	near	-	52
$S_{\text{req}} \setminus S_{\text{inf}}$	address	113	163
	phone	113	163
	postcode	113	163
	signature	113	163
	price	-	163

Table A.1: Slots in the restaurant information and tourist information domains. All the informable slots are also requestable. The second group, $S_{\text{req}} \setminus S_{\text{inf}}$, shows the requestable slots that are not informable. Note the *type* slot in the restaurant information domain has one value, and is always equal to *restaurant*. The possible values for *type* in the tourist information domain are $V_{\text{type}} = \{\text{restaurant}, \text{pub}, \text{coffee shop}\}$.

APPENDIX B

DIALOGUE ACT FORMAT

A dialogue act is a shallow representation of the semantics of either a user’s utterance or the system’s prompt. In the case of the system prompt, the dialogue act is input to a Natural Language Generation component, which generates text to be synthesised. Multiple formats have been proposed and are used for representing dialogue acts (Traum, 2000). This appendix describes the dialogue act format used in the Cambridge University Dialogue Systems group (Young, 2007), which is a relatively general format for representing the semantics of slot-based task driven dialogs.

Consider a dialogue domain that has requestable slots S_{req} and informable slots S_{inf} , and write $S = S_{req} \cup S_{inf}$. Let V_s denote the set of possible values for a slot $s \in S$. (Appendix A explains this terminology and describes the domains studied in this thesis).

A dialogue act consists of two components – a dialogue act type, $d\text{-type}$, and a set, X of *slot bindings*. There are three types of slot bindings:

Bound slots:	slots bound to values,	written “ $s = v$ ”
Negated bound slots:	slots bound to values and negated,	written “ $s \neq v$ ”
Unbound slots:	slots not bound to any value,	written “ s ”

where each slot s is in S , and slots are bound to values $v \in V_s$.

Consider an example dialogue act with $d\text{-type} = request$ and $X = \{phone, address, food = indian, area = west\}$. A full description of all dialogue acts follows, but this particular act corresponds to an utterance like, “What is the address and phone number of somewhere in the West serving Indian food?” This dialogue act is written in shorthand notation as follows: $request(phone, address, food = indian, area = west)$. The unbound slots in this act are *phone* and *address*.

Different dialogue act types impose restrictions on what must be contained in X , and are used differently depending on whether they are a user act or a system prompt. Ta-

bles B.1 and B.2 describe the dialogue acts for user utterances and system prompts respectively, grouped by the restrictions imposed on X . Table B.3 clarifies the formal definitions by giving examples of dialogue acts and corresponding English sentences.

Restrictions on X	Act Type	Description
X must be empty.	<i>bye</i>	Saying good-bye.
	<i>thankyou</i>	Saying thank you.
	<i>null</i>	An act that is not understandable by the system.
	<i>ack</i>	A back-channel such as ‘okay’ or ‘uh-huh’.
	<i>repeat</i>	Requesting the system to repeat itself.
	<i>restart</i>	Requesting the system start over.
X may contain bound slots and negated bound slot. X may not contain any unbound slots. All bound slots must be informable (in S_{inf}).	<i>help</i>	Asking for help.
	<i>inform</i>	The bound slots are interpreted as constraints from the user in the search for an entity. The user is informing the system that an entity is required whose attributes match the bound slots in X . For example <i>inform(food=chinese, area \neq west)</i> might correspond to “ <i>I want a chinese place that is not in the west.</i> ”
	<i>hello</i>	As with <i>inform</i> , but also greeting the system.
	<i>affirm</i>	As with <i>inform</i> , but also replying affirmatively to the last system prompt.
X must contain at least one unbound slot, and may optionally contain bound slots or negated bound slots involving informable slots.	<i>negate</i>	As with <i>inform</i> , but also replying negatively to the last system prompt.
	<i>request</i>	Asking the value of the unbound slots for an entity that currently matches the user’s goal, and also optionally giving constraints as in <i>inform</i> .

Table B.1: Description of dialogue act types for user actions. A slot binding is allowed, *this* = *Dontcare*, to give the constraint that the user does not care about the slot s , where s must be inferred from the last system prompt.

Restrictions on X	Act Type	Description
X must be empty.	<i>hello</i>	Giving a welcome message to initiate the dialogue.
	<i>reqmore</i>	Asking if the user needs any more information.
	<i>canthear</i>	Informing the user that they cannot be heard.
X may contain bound slots but may not contain any unbound slots.	<i>inform</i>	Offering an entity and informing its slot attributes as given in X .
$X = \{s = v\}$, i.e. X contains just one bound slot.	<i>confirm</i>	Confirming that the user wants a venue matching the constraint given by the bound slot.
$X = \{s\}$, i.e. X contains just one unbound slot.	<i>request</i>	Requesting what the user wants for the unbound slot.
X contains at least one bound slot and exactly one unbound slot.	<i>confreq</i>	Implicitly confirming the constraints given by the bound slots, and requesting what the user wants for the unbound slot.
$X = \{s = v_0, s = v_1\}$, i.e. X contains exactly two bindings for one slot.	<i>select</i>	Asking the user to select between the two suggested values for the slot.

Table B.2: Description of dialogue act types for system prompts. An optional special slot binding *count* = n is allowed for the *confreq* action to specify that there are n venues that match the other slot bindings. A special binding *name* = *none* is allowed for the *inform* action to inform the user there is no entity matching the given slot bindings. Another binding *other* = *true* is used in conjunction with this to further specify that there is no matching entity other than those mentioned.

User actions

<i>inform(area=east, food=Dontcare)</i>	I'm looking for something in the East of town serving any type of food.
<i>inform(this=Dontcare, pricerange=cheap)</i>	I don't mind but it should be cheap.
<i>inform(food=chinese, food≠thai)</i>	No, I want Chinese food, not Thai.
<i>hello(area=north)</i>	Hi, do you have a place in the North?
<i>negate(area=Dontcare)</i>	No, in any part of town.
<i>affirm(area=Dontcare, food=thai)</i>	Yes, Thai food anywhere.
<i>request(phone, address)</i>	What's their address and phone number?
<i>request(phone, area=north)</i>	Do you know the phone number of one in the North?

System prompts

<i>hello()</i>	Hello, welcome to the Cambridge restaurant information system. You can ask for restaurants by area, pricerange or food type. How may I help you?
<i>reqmore()</i>	Can I help you with anything else?
<i>inform(name=cocum, area=west, food=indian)</i>	Cocum is a nice place serving Indian food in the West of town.
<i>inform(name=cocum, address='71 Castle Street', postcode='CB3 0AH')</i>	Sure, Cocum is on 71 Castle Street and its postcode is CB3 0AH.
<i>inform(name=none, area=west, food=chinese)</i>	I'm sorry there is no restaurant serving Chinese food in the West of town.
<i>inform(name=none, other=true, food=japanese)</i>	There is no other restaurant that serves Japanese food.
<i>confreq(food=indian, area)</i>	Okay, an Indian place. What part of town did you have in mind?
<i>confreq(food=indian, pricerange=Dontcare, count=13, area)</i>	There are 13 Indian restaurants if you don't care about the pricerange. What part of town would you like?
<i>request(food)</i>	What type of food would you like?
<i>confirm(food=indian)</i>	You would like Indian food, is that right?
<i>select(food=indian, food=italian)</i>	I'm sorry, would you like Indian or Italian food?

Table B.3: Examples of dialogue acts with corresponding realisations in English, in the Cambridge restaurant information domain.

APPENDIX C

MATCHING STRINGS TO THE DOMAIN WITH ALIASES

Sequential labelling techniques for Spoken Language Understanding (SLU), such as the Conditional Random Field (CRF) model presented in section 3.1.2, require alignments between words in the input sentence and the slot values and slot names in the dialogue act. It is also necessary to identify occurrences of slot values and slot names in the input sentences for creating *delexicalised* features in Dialogue State Tracking (DST) (see section 5.6).

This thesis uses a simple approach to find occurrences in the input sentence string of slot names and slot values by exploiting sets of *aliases*. A slot name or slot value is identified in the input string if it exactly matches the string as it appears in the domain specification, or one of the aliases specified by the system designer. For example, the slot value *area = east* would be identified in the sentence “*Something in the east*”, as the string *east* exactly matches the slot value in the domain specification. If *eastern* is chosen as an alias for *area = east*, then similarly this slot value would be identified in the sentence “*Something in the eastern part*”.

C.1 Aliases

This section presents the sets of aliases used in this thesis both for labelling the SLU data of section 3.2.1, and creating delexicalised features in section 6.3.

C.1.1 Aliases for DSTC 3 evaluation

The following aliases were used for training delexicalised Recurrent Neural Network (RNN) trackers on the Dialog State Tracking Challenge (DSTC) 2 data in the restaurant information

domain, which are then applied to the DSTC 3 data in the tourist information. This is an example of training on one domain, and testing on an expanded domain that contains several new slots. A full specification of these domains is given in appendix A. Using aliases in this context allows the trackers to more readily identify slot names and slot values in the expanded domain without requiring in-domain training data.

These aliases are generated from the Phoenix grammar (Ward, 1994) used in the data collection for DSTC 3. The aliases consist of alternative expressions such as *children allowed* and *children permitted*. Many of the aliases are shorter versions of the phrases that appear in the domain specification, such as *tex mex* for *mexican tex mex*.

The list below omits slots and values for which no aliases are used.

Slot names

- area: location, part, part of town, region, side
 - mexican tex mex: mexican, tex mex
 - north american: american
- food: cuisine, kind of food, type of food
- name: called
- near: a view of, a view over, adjacent, around, close to, closer to, facing, near to, next to, opposite, part of
 - true: broadband, broadband access, broadband connection, internet, internet access, internet connection, wifi, wifi access, wifi connection
- phone: number, phone number, telephone number
- postcode: postal code, zip code
- pricerange: price, price range
- signature: signature dish
- has internet:
- has tv:
 - true: has a television, has a tv, has television, televison, tv

Slot values

- area:
 - anatolia turkish restaurant: anatolia
 - arundel house hotel: arundel
 - b bar and restaurant: b bar
 - browns restaurant: browns
 - carringtons cafe restaurant: carringtons
 - centre: central, city centre, middle
 - riverside: river, river cam, river side, the cam
- children allowed:
 - chiquito mexican bar and grill: chiquito
 - courtyard cafe at the fitzwilliam museum: courtyard cafe
 - false: children forbidden, kids forbidden
 - de luca cucina and bar: de luca
 - dojo noodle bar: dojo
 - true: children, children permitted, children welcome, kids, kids allowed, kids permitted, kids welcome
- name:

- don pasquale restaurant: don pasquale
 - edwinns bar and restaurant: edwinns
 - efes restaurant: efes
 - fitzbillies restaurant: fitzbillies
 - hotel du vin and bistro: hotel du vin
 - kymmoy noodle bar and restaurant: kymmoy
 - la margherita: margherita
 - loch fyne restaurant: loch fyne
 - lucky star chinese buffet restaurant: lucky star
 - maharajah restaurant: maharajah
 - milton park english and thai restaurant: milton park
 - panahar tandoori restaurant: panahar tandoori
 - peking restaurant: peking
 - pipasha restaurant: pipasha
 - rainbow vegetarian cafe: rainbow cafe
 - river bar and kitchen: river bar
 - royal cambridge hotel: royal cambridge
 - sala thong restaurant: sala thong
 - sesame restaurant and bar: sesame
 - spice merchants indian restaurant: spice merchants
 - stazione restaurant and coffee bar: stazione
 - tang chinese: tang
 - the agora at the copper kettle: agora
 - the alma: alma
 - the anchor: anchor
 - the avery: avery
 - the bakers: bakers
 - the baron of beef: baron of beef
 - the bombay brasserie: bombay brasserie
 - the cambridge chop house: cambridge chop house
 - the cow: cow
 - the curry house: curry house
 - the curry king: curry king
 - the curry queen tandoori: curry queen
 - the eagle: eagle
 - the fleur bar and bistro: fleur bar and bistro
 - the fountain inn: fountain inn
 - the gandhi: gandhi
 - the golden curry: golden curry
 - the golden palace: golden palace
 - the kohinoor tandoori restaurant: kohinoor tandoori
 - the lion and lamb: lion and lamb
 - the oak bistro: oak bistro
 - the old crown: old crown
 - the phoenix chinese restaurant: phoenix
 - the punter: punter
 - the red bull: red bull
 - the rice boat: rice boat
 - the saffron brasserie: saffron brasserie
 - the sorrento hotel and restaurant: sorrento
 - the tram depot: tram depot
 - the unicorn steak and ale house: unicorn steak and ale house
 - the vaults: vaults
 - the wrestlers: wrestlers
 - varsity restaurant: varsity
 - yippee noodle bar: yippee
- near:
 - broughton house gallery: broughton house
 - cambridge and county folk museum:

Appendix C. Matching Strings to the Domain with Aliases

- county folk museum, folk museum
- cambridge artworks: art space, art works, artworks
- cambridge book and print gallery: book and print gallery
- cambridge contemporary art: contemporary art
- cambridge museum of technology: museum of technology, technology museum
- cambridge university botanic gardens: botanic garden, botanic gardens, botanical garden, botanical gardens
- churchill college: churchill
- magdalene college: magdelene
- museum of classical archaeology: museum of archaeology
- pembroke college: pembroke
- sheeps green and lammas land park fen causeway: lammas land, lammas land park, sheeps green
- sidney sussex college: sidney sussex
- whipple museum of the history of science: whipple museum
- pricerange:
 - cheap: basic, cheapest, inexpensive, simple
 - expensive: luxurious, posh, upmarket
 - moderate: average price, average priced, average pricerange, average range, medium price, medium priced, medium pricerange, medium range, mid price, mid priced, mid pricerange, mid range, moderate price, moderate pricerange, moderate range, moderately priced, reasonable price, reasonable priced, reasonable pricerange, reasonable range, reasonably priced
- type:
 - coffeeshop: cafe, cafes, coffee, coffee shop, coffee shops
 - pub: pubs
 - restaurant: place to eat, restaurants

C.1.2 Supplementary Aliases for CRF Tagging

The set of aliases presented in section C.1.1 is supplemented with a small set of additional aliases for labelling the SLU corpus of section 3.2.1. These additional aliases were chosen by iterating on the SLU data until all alignments were found.

These aliases include alternative phrasings for the *Dontcare* value, which means that the user does not wish to constrain a particular slot. Note that no delexicalisation is done for the *Dontcare* value in RNN-based DST, so these are not necessary in the aliases of section C.1.1.

Slot names

- address: where, where is
- food: serve, serves

whichever

- area:
 - Dontcare: anywhere
 - east: eastern
 - north: northern
 - south: southern
 - west: western

Slot values

- *All slots:*
 - Dontcare: any, anything, do not care, doesn't matter, don't care, don't mind, i don't know, whatever,
- food:
 - asian oriental: asian, oriental

APPENDIX D

LEARNING NEURAL NETWORKS

Artificial neural networks define complex functions of input streams that can be learnt from data. Inspired by biological neural networks, they are composed of interconnected *neurons* whose activations are dependent on the activations of their inputs. This appendix briefly presents the definition of the Multi-layer Perceptron (MLP) and the concept of Recurrent Neural Networks (RNNs), and discusses how these can be learnt from data.

The first work on artificial neural networks used electrical circuits to model how small networks of biological neurons might work (McCulloch and Pitts, 1943). The 1980s saw a revival of interest in these models, with e.g. a paper presented to the national Academy of Sciences by Hopfield (1982). By 1985 the American Institute of Physics had begun *Neural Networks for Computing*, now an annual meeting. Since then there has been an explosion in research on neural network techniques and applications.

Fuelled by advances in computer architectures, the availability of data and new learning techniques, there has more recently been another burst in interest applying neural networks in machine learning, particularly for applications in speech and language technology (Deng et al., 2013; Hinton et al., 2012).

D.1 Multi-layer Perceptrons

The Multi-layer Perceptron (MLP) approximates a function from $\mathbb{R}^{n_{\text{input}}} \rightarrow \mathbb{R}^{n_{\text{output}}}$, i.e. from real-valued vectors of size n_{input} to real-valued vectors of size n_{output} . The function is written $NNet(\cdot)$.

The input vector, $\mathbf{x} \in \mathbb{R}^{n_{\text{input}}}$ is mapped into a series of m *hidden layers*, and the last hidden layer is mapped into the output $\mathbf{y} \in \mathbb{R}^{n_{\text{output}}}$. The hidden layers are simply vectors, $\mathbf{h}_0, \dots, \mathbf{h}_{m-1}$, whose components represent the activation levels of neurons in the model. The mapping between layers models biological neural connections, where the activation for

each *neuron* is a function of a weighted sum of the neurons it is connected to in the previous layer.

The MLP function, $\mathbf{y} = \text{NNet}(\mathbf{x})$, is as follows:

$$\begin{aligned}\mathbf{h}_0 &= g_0(W_0 \mathbf{x}^T + b_0) \\ \mathbf{h}_i &= g_i(W_i \mathbf{h}_{i-1}^T + b_i) \quad \text{for } 0 < i < m \\ \mathbf{y} &= g_m(W_m \mathbf{h}_{m-1}^T + b_m)\end{aligned}\tag{D.1}$$

where the W_i weight matrices and b_i bias vectors are parameters of the model. The dimensionality of the hidden layers must be chosen by the system designer. The g_i activation functions are differentiable functions from $\mathbb{R}^n \rightarrow \mathbb{R}^n$. Commonly used activation functions are:

- \tanh , the hyperbolic tangent, with $\tanh(\mathbf{h})_i = e^{h_i - e^{-h_i}} / e^{h_i + e^{-h_i}}$. This most commonly used activation function maps the activations to the range $(-1, 1)$.
- σ , the sigmoid function, defined as $\sigma(\mathbf{h})_i = e^{h_i} / (1 + e^{h_i})$. This is similar to \tanh but maps to the range $(0, 1)$.
- softmax, a function whose output is a categorical probability distribution, $\text{softmax}(\mathbf{h})_i = e^{h_i} / \sum_j e^{h_j}$

The final output activation g_m must be chosen so that the outputs lie in the correct range for the function being approximated. For example if a categorical probability distribution is required, a softmax output activation should be chosen.

Note that the final composed function $\text{NNet}(\mathbf{x})$ is differentiable with respect to all the parameters W_i and b_i . This means that it can be learnt using gradient descent methods to optimise a cost function of the output \mathbf{y} , see section D.4.

D.2 Parameter Initialisation

Prior to optimising the parameters using gradient descent methods, it is important to initialise the parameters well. Two methods are used for initialisation in this thesis, random initialisation and denoising Autoencoder (dA) initialisation.

D.2.1 Random Initialisation

With random initialisation, the components of the W_i and b_i parameters are sampled from uniform random distributions in the range $(-1, 1)$. When using \tanh activations, Glorot and

Bengio (2010) recommend scaling the components of each of the weight matrices W_i by a factor of

$$\sqrt{\frac{6}{n_{rows} + n_{cols}}} \quad (\text{D.2})$$

where n_{rows} and n_{cols} are the number of rows and columns of W_i . This ensures that the expected initial activations of the neurons is within a reasonable domain for the activation function. This type of initialisation is used in this thesis unless otherwise specified.

D.2.2 Denoising Autoencoder Initialisation

Parameters can also be initiated to informative values using a denoising Autoencoder (dA). The definition given here applies to an MLP with one hidden layer, but can be generalised to multiple layers. This unsupervised method for learning meaningful underlying representations of the input has been found effective as an initialisation technique in deep learning research (Vincent et al., 2008).

The dA learns a matrix W_{dA} , which reduces the input \mathbf{x} to a lower dimensional hidden layer vector such that the original vector may be recovered with minimal loss in the presence of noise.

For learning the dA, \mathbf{x} is first normalised such that feature values lie between 0 and 1. The dA takes as input \mathbf{x}_{noisy} , a noisy copy of \mathbf{x} where each component is set to 0 with probability p . This is mapped to a lower dimensional hidden representation \mathbf{h} :

$$\mathbf{h} = \sigma(W_{dA}^T \mathbf{x}_{noisy} + \mathbf{b}_{dA,0}) \quad (\text{D.3})$$

A reconstructed vector, \mathbf{x}_{rec} , is then calculated as:

$$\mathbf{x}_{rec} = \sigma(W_{dA} \mathbf{h} + \mathbf{b}_{dA,1}) \quad (\text{D.4})$$

The cross-entropy between \mathbf{x} and \mathbf{x}_{rec} , $H(\mathbf{x}, \mathbf{x}_{rec}) = -\sum_i x_i \log x_{rec,i}$, is used as the objective function in gradient descent to optimise W_{dA} , $\mathbf{b}_{dA,0}$ and $\mathbf{b}_{dA,1}$. In dA initialisation, the first layer of the MLP is initialised with the parameters W_{dA} and $\mathbf{b}_{dA,0}$.

D.3 Recurrent Neural Networks

Recurrent Neural Networks (RNNs) are neural networks that take a series of inputs \mathbf{x}_t and produce a series of outputs \mathbf{y}_t for $t = 0, \dots, T - 1$. Connections between the hidden layers

in adjacent time steps allow the network to model sequences. This section will describe two of the most commonly used recurrent neural network structures, Jordan and Elman RNNs.

An Elman-type RNN (Elman, 1990) has recurrent connections between hidden layers of the neural network. The outputs \mathbf{y}_t are calculated as an MLP of the inputs \mathbf{x}_t , so $\mathbf{y}_t = \text{NNet}(\mathbf{x}_t)$ but the equation for the j^{th} hidden layer $\mathbf{h}_{j,t}$ is altered from the original given in equation (D.1):

$$\mathbf{h}_{j,t} = g_j(W_j^T \mathbf{h}_{j-1,t} + \mathbf{b}_j) \quad (\text{D.5})$$

to:

$$\mathbf{h}_{j,t} = g_j(W_j^T \mathbf{h}_{j-1,t} + W_{\text{rec},j}^T \mathbf{h}_{j,t-1} + \mathbf{b}_j) \quad (\text{D.6})$$

which introduces a recurrent connection, i.e. a dependency on the previous time step.

A Jordan-type RNN (Jordan, 1986) feeds the output of one time-step into the network for the following time step. The equation for the first hidden layer becomes:

$$\mathbf{h}_{0,t} = g_0(W_0^T \mathbf{x}_t + W_{\text{rec}} \mathbf{y}'_{t-1} + \mathbf{b}_0) \quad (\text{D.7})$$

where \mathbf{y}'_t are the outputs of the RNN.

Given an example sequence $(\mathbf{x}_t, \mathbf{y}_t)$, both types of RNN can be unrolled to give a single MLP-type network with T outputs. The parameters of this network can then be optimised with standard gradient descent methods. This technique is termed *back propagation through time* (Mozier, 1988).

Back propagation through time can lead to a problem with vanishing or exploding gradients when using sigmoid or tanh activation functions. One simple method for avoiding this problem is to *clip* the gradient calculations so they lie in a reasonable range. Other RNN structures have been developed that avoid this problem, such as long short term memory networks (Hochreiter and Schmidhuber, 1997).

D.4 Stochastic Gradient Descent

When training an MLP, $\mathbf{y} = \text{NNet}(\mathbf{x})$, assume there is a list of example inputs and outputs \mathbf{x}_i and \mathbf{y}_i . Let $\mathbf{y}'_i = \text{NNet}(\mathbf{x}_i)$, the estimation of \mathbf{y}_i given by the MLP. To train this network using Stochastic Gradient Descent (SGD), a cost function is required to compare \mathbf{y}' with \mathbf{y} for any set of parameters $\{W_j\}$ and $\{b_j\}$:

$$\text{cost}(\mathbf{y}'_i, \mathbf{y}_i, \{W_j\}, \{b_j\}) \quad (\text{D.8})$$

In the case of estimating a probability distribution, the target output vectors \mathbf{y}_i are delta distributions, i.e. \mathbf{y}_i is identically 0 except in the position of the correct index where its value is 1. The cost might then be the negative log probability of the true label:

$$-\sum_j y_{i,j} \log y'_{i,j} \quad (\text{D.9})$$

Gradients of the cost function with respect to each of the parameters θ (W_i or b_i) can then be computed, $\frac{d}{d\theta} \text{cost}$. In this work it has been useful to exploit methods for automatically calculating the gradients for arbitrary neural network structures such as *Theano* (Bergstra et al., 2010).

SGD uses small batches of m training examples to update the parameters by moving in the direction of the gradient calculated using each batch. It works as follows:

1. Initialise $\{W_j\}$ and $\{b_j\}$ using e.g. a technique in section D.2
2. Select m random training examples, call these \mathbf{x}_i and \mathbf{y}_i for $i = 0, \dots, m-1$.
3. For each parameter θ , a scalar component of W_j or b_j for some j , let $g_\theta = 0$.
4. For each example $\mathbf{x}_i, \mathbf{y}_i$ and parameter θ :
 - (a) Calculate the gradient of the cost with respect to θ for this training example.

$$g_\theta^i = \frac{d}{d\theta} \text{cost}(\mathbf{y}'_i, \mathbf{y}_i, \{W_j\}, \{b_j\})$$

- (b) Optionally clip the gradient so that it lies in a reasonable range. This is particularly important for models with many layers or RNNs.

$$g_\theta^i \leftarrow \max\{-G, \min\{G, g_\theta^i\}\}$$

where $G > 0$.

- (c) Update the gradient calculation for the batch of examples:

$$g_\theta \leftarrow g_\theta + g_\theta^i$$

5. Update each parameter θ , by moving in the direction of decreasing cost according to the gradient calculation:

$$\theta \leftarrow \theta - \eta g_\theta$$

where $\eta > 0$ is the *learning rate*.

6. Return to step 2.

This procedure can either run for some fixed number of loops, or can continue until learning has converged. One typical heuristic used to decide whether learning has converged is to always train for n loops, but consider increasing n at step 6 if the error on a set of held out examples improves by a certain threshold.

D.5 Regularisation

When learning neural networks, it is useful to include a term in the cost function that penalises large parameters, called a regularisation term. Learning parameters with regularisation is observed to improve generalisation of the learnt network. Preferring smaller weights can be regarded as preferring simpler models.

The regularisation term for a neural network with parameters $\{W_i\}$ and $\{b_i\}$ is:

$$\lambda \sum_i (|W_i| + |b_i|) \tag{D.10}$$

where $\lambda > 0$ is a scaling factor, and $|\cdot|$ is typically either the l1 or l2 norm of the vector or matrix. This work uses l2 regularisation.

GLOSSARY

accuracy

evaluation statistic for dialogue state tracking. How often the 1-best hypothesis is correct. 44, 61, 63–65, 73, 76, 77, 79, 80, 95–97, 100, 104

alias

an alternative phrasing for a slot name (such as ‘area’) or slot value (such as ‘east’). 24, 25, 28, 60, 66, 74, 77, 111, 123, 124, 127

Bayesian network

a directed acyclic graph whose nodes are random variables. The joint distribution of the variables decomposes into factors of the form $P(v | pa(v))$ where $pa(v)$ is the set of parents of v . 47, 75, 77–83, 86, 91, 100, 104

belief

the distribution a dialogue state tracker reports over a particular component of the dialogue state. 11, 14, 44

BIO tag

Beginning Inside or Outside tag. An encoding for labelling sequences. 9, 10, 18–20, 24

CNet decoder

semantic decoder using the Semantic Tuple Classifier method, utilising features extracted from the word confusion network and last machine action. 3, 30, 34–40, 83, 109, 111

decoder

a spoken language understanding component, which converts sentences in natural language to a dialogue act format. 8, 18, 22, 23, 25–31, 33–35, 40

delexicalised

(to describe an input utterance to a dialogue state tracker) an utterance where words representing possible slot values in the domain have been replaced with generic tags. A *delexicalised* dialogue state tracker is one that operates on such input. 55, 57, 58, 61, 66, 77, 83, 90, 95, 96, 109, 111, 113, 123

Dialog State Tracking Challenges

a series of research challenges in dialogue state tracking. Spelling of *dialog* follows the official name. 4, 11, 30, 42, 67, 68, 81, 94, 99, 109, 123, 139

dialogue reward

a real number score assigned to a dialogue, such that higher scores correspond to better dialogues. This is the metric optimised when learning a dialogue policy during reinforcement learning. 4, 25, 30, 38, 39, 84–86

discriminative

to describe a probabilistic model that directly models the conditional probability of the outputs of a system given the inputs. 2–4, 6, 9–12, 17, 20, 22, 27, 28, 30, 41, 50, 51, 53, 64, 67, 76, 80, 81, 90, 100, 109, 111, 112

dynamic Bayesian network

a Bayesian network that relates random variables to each other over adjacent timesteps. 10–12, 48, 49, 66, 69, 76, 91

F-score

geometric mean of the recall and precision of semantic items in the top hypothesis from a spoken language understanding component. 25–27, 33–35, 37, 38, 40, 71, 72

generative

to describe a probabilistic model that jointly models the inputs and required outputs of a system. Often the required outputs are modelled as hidden random variables, which can be inferred from observing the inputs. 3, 9–12, 15, 20, 47, 50, 64, 67, 76, 79, 81–83, 91, 109

goal constraint

a constraint for a particular slot that the user has given. 42–44, 46–48, 51, 55, 59–61, 63–66, 73, 76, 77, 79, 80, 94–97, 99, 100, 102, 104, 106

Item Cross Entropy

the cross entropy between the true semantic labels and the reported probabilities from a spoken language understanding component. 25, 33, 139

L2

evaluation statistic for dialogue state tracking. The square of the L2 norm between the reported dialogue state distribution and the true (delta) label distribution. 61, 63, 73, 76, 77, 79, 80, 90, 95–97, 100

N-best list

the top N scoring hypotheses from the automatic speech recognition. 6, 7, 13, 18, 26–32, 34, 36, 61, 63, 74, 83

policy

a mapping from the dialogue state, or distribution over dialogue states, to the next system action. 14, 42, 69, 71, 81, 83–85, 89

requested slot

a slot that has been requested by the user, and must be informed by the system. 42–44, 46–48, 59, 60, 68, 73, 76

search method

the method by which the user is trying to interact with the dialogue system at a given point in the dialogue. This is one of *by constraints*, *by name*, *by alternatives* or *finished*. 42–48, 59, 60, 68, 73, 76, 112

turn

a dialogue turn, i.e. one system prompt and user utterance. 5, 12–14, 38–44, 46, 47, 51, 53, 55–57, 62, 73, 81, 87–90, 92, 93, 97, 112

word confusion network

representation of the full posterior distribution from the automatic speech recognition component. 4, 6–8, 11, 13, 28–32, 35, 40, 61, 74, 76, 109, 112

word-based

(to describe a dialogue state tracker) directly operates on the output of the speech recognition, with no explicit intermediate semantic spoken language understanding step. 4, 41, 60, 61, 63, 65, 66, 74, 76, 77, 81, 83, 86, 91, 95, 96, 104, 105, 109, 111, 113

a

the last action taken by the system, a dialogue act. 45, 48, 49

 \oplus

the concatenation operator. If **a** and **b** are vectors, then **a** \oplus **b** is the concatenation of **a** and **b**, i.e. its components are all of the components of **a** followed by all the components of **b**. 55, 56, 59, 101–103

d-type

the type of a dialogue act. 22, 23, 25, 117

NNet

a Multi-layer Perceptron function. 56, 129, 130, 132

 o_{gs}

the observation from the SLU of the goal constraint for slot *s*. This is a vector giving a value between 0 and 1 for each possible value for the slot, summing to at most 1. 44, 49

o_i

the observation of component i of the dialogue state given by the SLU. 44–46, 48, 73

o_m

the observation from the SLU of the search method. This is a vector giving a value between 0 and 1 for each possible method (including *none*), summing to 1. 44, 49

o_{r_s}

the observation from the SLU of the slot s being requested. A scalar between 0 and 1. 44, 49

S

the set of all slots in a slot-based dialogue domain. 115, 117

S_{inf}

the set of informable slots. 23, 42, 44, 46, 49, 55, 56, 115–117, 119

S_{req}

the set of requestable slots. 42, 44, 46, 47, 49, 59, 115–117

V_s

the set of possible values for slot s . 23, 42, 44–46, 55, 56, 115, 117

X

the set of slot-bindings belonging to a dialogue act. 22, 25, 117–120

ACRONYMS

ASR

Automatic Speech Recognition. 4, 6–8, 13, 16, 21, 22, 24, 25, 27–30, 32, 33, 38, 39, 49–51, 57, 58, 60, 63, 68–70, 72–77, 80, 82, 83, 87, 90, 92, 99, 103

ATIS

Air Travel Information System. 9, 20

BUDS

Bayesian Update of Dialogue State. 11, 13, 34, 40, 46, 80, 96

CRF

Conditional Random Field. 10, 15–22, 24–26, 28, 31–33, 51, 52, 75, 115

dA

denoising Autoencoder. 60, 122, 123

DST

Dialogue State Tracking. 3, 4, 6, 10–13, 39, 40, 42, 45–47, 49, 51, 52, 56–58, 62, 63, 65, 66, 71, 72, 77, 79, 80, 82, 86, 96, 101, 103, 104, 115, 119

DSTC

Dialog State Tracking Challenge. 4, 10, 11, 28, 40, 42, 46, 58, 60, 62, 65–76, 78–80, 89–92, 95, 96, 99, 103, 115, 116, *Glossary*: Dialog State Tracking Challenges

HMM

Hidden Markov Model. 6

ICE

Item Cross Entropy. 23–25, 31–33, 35, 36, *Glossary*: Item Cross Entropy

MDP

Markov Decision Process. 12, 14

MLP

Multi-layer Perceptron. 121–124

NLG

Natural Language Generation. 14

POMDP

Partially Observable Markov Decision Process. 12, 13, 34, 67, 70, 80, 101

RNN

Recurrent Neural Network. 4, 6, 14, 17, 39, 42, 49, 52, 54, 57, 58, 60–63, 65, 71–77, 79, 80, 82, 86–88, 90–93, 95, 98–101, 103–105, 115, 119, 121, 123, 124

SDS

Spoken Dialogue System. 3, 5, 10, 12, 23, 46

SGD

Stochastic Gradient Descent. 58, 60, 87, 90, 92, 93, 98, 124, 125

SLU

Spoken Language Understanding. 3, 4, 6, 7, 9–13, 15–17, 21, 23–25, 27, 28, 30, 31, 39, 41, 42, 44–46, 48–53, 55, 57, 58, 62, 68, 70–80, 86, 96, 99, 100, 103, 104, 115, 119

SS

Speech Synthesis. 14

STC

Semantic Tuple Classifier. 3, 10, 15, 19–21, 24, 25, 28, 30–32, 38

SVM

Support Vector Machine. 3, 10, 19, 20, 30, 31

WER

Word Error Rate. 21, 27, 33, 35–37, 67–70, 80

REFERENCES

- A. Acero and Y. Y. Wang, “Spoken language understanding,” *IEEE Signal Processing Magazine*, vol. 22, no. 5, Sep. 2005. 10
- T. Becker, N. Blaylock, C. Gerstenberger, I. Kruijff-Korbayová, A. Korthauer, M. Pinkal, M. Pitz, P. Poller, and J. Schehl, “Natural and intuitive multimodal dialogue for in-car applications: The SAMMIE system,” *FRONTIERS IN ARTIFICIAL INTELLIGENCE AND APPLICATIONS*, vol. 141, p. 612, 2006. 1
- S. Bengio and Y. Bengio, “Taking on the curse of dimensionality in joint distributions using neural networks,” *Neural Networks, IEEE Transactions on*, vol. 11, no. 3, May 2000. 101, 106
- J. Bergstra, O. Breuleux, F. Bastien, P. Lamblin, R. Pascanu, G. Desjardins, J. Turian, D. Warde-Farley, and Y. Bengio, “Theano: a CPU and GPU math expression compiler,” in *Proceedings of the Python for Scientific Computing Conference (SciPy)*, Jun. 2010, oral Presentation. 133
- A. Biermann and P. M. Long, “The composition of messages in speech-graphics interactive systems,” in *In Proceedings of the 1996 International Symposium on Spoken Dialogue*, 1996, pp. 97–100. 13
- C. M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2006. 47
- A. Black, S. Burger, A. Conkie, H. Hastie, S. Keizer, O. Lemon, N. Merigaud, G. Parent, G. Schubiner, B. Thomson, J. Williams, K. Yu, S. Young, and M. Eskenazi, “Spoken dialog challenge 2010: Comparison of live and control test results,” in *SIGdial*. Association for Computational Linguistics, 2011. 68
- D. Bohus, “Error Awareness and Recovery in Conversational Spoken Language Interfaces,” Ph.D. dissertation, Carnegie Mellon University, 2007. 2
- D. Bohus and A. Rudnicky, “Sorry, I didn’t catch that! An investigation of non-understanding errors and recovery strategies,” in *SIGdial*. Association for Computational Linguistics, 2005. 13
- , “A K-hypotheses + Other Belief Updating Model,” 2006. 12, 51
- H. A. Bourlard and N. Morgan, *Connectionist Speech Recognition: A Hybrid Approach*. Norwell, MA, USA: Kluwer Academic Publishers, 1993. 6

References

- Y.-N. Chen, W. Y. Wang, and A. I. Rudnicky, “An empirical investigation of sparse log-linear models for improved dialogue act classification,” in *International Conference on Acoustics, Speech, and Signal Processing*. IEEE, 2013, pp. 8317–8321. 24
- Y.-N. Chen, W. Wang, and A. Rudnicky, “Unsupervised induction and filling of semantic slots for spoken dialogue systems using frame-semantic parsing,” in *Workshop on Automatic Speech Recognition and Understanding (ASRU)*. IEEE, Dec 2013, pp. 120–125. 24
- Y.-N. Chen, W. Y. Wang, and A. I. Rudnicky, “Leveraging frame semantics and distributional semantics for unsupervised semantic slot induction in spoken dialogue systems,” in *Spoken Language Technology Workshop*. IEEE, 2014. 24, 111
- P. R. Cohen and S. L. Oviatt, “The role of voice input for human-machine communication,” in *Proceedings of the National Academy of Sciences*, 1994, pp. 9921–9927. 1
- D. Dahl, M. Bates, M. Brown, W. Fisher, K. Hunicke-Smith, D. Pallett, C. Pao, A. Rudnicky, and E. Shriberg, “Expanding the scope of the ATIS task: The ATIS-3 corpus,” in *Proceedings of the workshop on Human Language Technology*. Association for Computational Linguistics, 1994, pp. 43–48. 10, 23
- R. de Mori, “Spoken language understanding: a survey,” in *Workshop on Automatic Speech Recognition and Understanding (ASRU)*. IEEE, 2007. 8
- L. Deng, J. Li, J.-T. Huang, J.-T. Yao, D. Yu, F. Seide, M. Seltzer, G. Zweig, X. He, J. Williams, Y. Gong, and A. Acero, “Recent advances in deep learning for speech research at microsoft.” *International Conference on Acoustics, Speech, and Signal Processing*, May 2013. 6, 129
- A. Deoras, R. Sarikaya, G. Tur, and D. Hakkani-Tur, “Joint decoding for speech recognition and semantic tagging,” in *Annual Conference of the International Speech Communication Association (INTERSPEECH)*, September 2012. 30
- A. Deoras, G. Tur, R. Sarikaya, and D. Hakkani-Tur, “Joint discriminative decoding of word and semantic tags for spoken language understanding,” *IEEE Transactions on Audio, Speech, and Language Processing*, 2013. 30
- J. Duchi, E. Hazan, and Y. Singer, “Adaptive subgradient methods for online learning and stochastic optimization,” *J. Mach. Learn. Res.*, vol. 12, Jul. 2011. 66
- J. L. Elman, “Finding structure in time,” *COGNITIVE SCIENCE*, vol. 14, no. 2, pp. 179–211, 1990. 132
- R. Epstein, “The quest for the thinking computer,” *AI Magazine*, vol. 13, no. 2, pp. 81–95, Jul. 1992. 1
- F. Faber, M. Bennewitz, C. Eppner, A. Gorog, C. Gonsior, D. Joho, M. Schreiber, and S. Behnke, “The humanoid museum tour guide robotinho,” in *Robot and Human Interactive Communication, 2009. RO-MAN 2009. The 18th IEEE International Symposium on*. IEEE, 2009, pp. 891–896. 1
- C. Fellbaum, *WordNet: An Electronic Lexical Database*. Bradford Books, 1998. 61

- S. Feng, R. Manmatha, and A. McCallum, “Exploring the Use of Conditional Random Field Models and HMMs for Historical Handwritten Document Recognition,” in *Proceedings of the Second International Conference on Document Image Analysis for Libraries*, ser. DIAL ’06. Washington, DC, USA: IEEE Computer Society, 2006, pp. 30–37. 28
- M. E. Foster, A. Gaschler, M. Giuliani, A. Isard, M. Pateraki, and R. P. Petrick, “Two people walk into a bar: Dynamic multi-party social interaction with a robot agent,” in *Proceedings of the 14th ACM International Conference on Multimodal Interaction*, ser. ICMI ’12. New York, NY, USA: ACM, 2012, pp. 3–10. 1
- M. Gales and S. Young, “The Application of Hidden Markov Models in Speech Recognition,” *Foundations and Trends in Signal Processing*, vol. 1, no. 3, pp. 195–304, 2007. 6
- J. Ganitkevitch, B. Van Durme, and C. Callison-Burch, “Ppdb: The paraphrase database,” in *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Atlanta, Georgia: Association for Computational Linguistics, June 2013, pp. 758–764. 112
- M. Gašić, F. Jurcicek, S. Keizer, F. Mairesse, B. Thomson, K. Yu, and S. Young, “Gaussian processes for fast policy optimisation of pomdp-based dialogue managers,” in *Proceedings of the 11th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, ser. SIGdial. Stroudsburg, PA, USA: Association for Computational Linguistics, 2010, pp. 201–204. 14, 106
- M. Gašić, P. Tsiakoulis, M. Henderson, B. Thomson, K. Yu, E. Tzirkel, and S. Young, “The effect of cognitive load on a statistical dialogue system,” in *SIGdial*. Association for Computational Linguistics, 2012. 24
- M. Gašić, C. Breslin, M. Henderson, D. Kim, M. Szummer, B. Thomson, P. Tsiakoulis, and S. Young, “On-line policy optimisation of Bayesian spoken dialogue systems via human interaction,” in *International Conference on Acoustics, Speech, and Signal Processing*. IEEE, 2013. 14, 84, 85
- X. Glorot and Y. Bengio, “Understanding the difficulty of training deep feedforward neural networks,” in *In Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS’10)*. Society for Artificial Intelligence and Statistics, 2010. 130
- D. Goddeau, H. Meng, J. Polifroni, S. Seneff, and S. Busayapongchai, “A form-based dialogue manager for spoken language applications,” in *Spoken Language, 1996. ICSLP 96. Proceedings., Fourth International Conference on*, vol. 2, Oct 1996. 13
- A. Graves and N. Jaitly, “Towards end-to-end speech recognition with recurrent neural networks,” in *ICML*, T. Jebara and E. P. Xing, Eds., 2014, pp. 1764–1772. 6
- A. Graves, G. Wayne, and I. Danihelka, “Neural turing machines,” *CoRR*, vol. abs/1410.5401, 2014. 112
- N. Gupta, G. Tur, D. Hakkani-Tur, S. Bangalore, G. Riccardi, and M. Gilbert, “The AT&T spoken language understanding system,” *Audio, Speech, and Language Processing, IEEE Transactions on*, pp. 213 – 222, 2006. 9

References

- T. J. Hazen, S. Seneff, and J. Polifroni, “Recognition confidence scoring and its use in speech understanding systems,” *Computer Speech & Language*, vol. 16, no. 1, pp. 49 – 67, 2002. 29
- Y. He and S. Young, “Spoken language understanding using the Hidden Vector State Model,” *Speech Communication*, vol. 48, no. 3-4, pp. 262–275, Mar. 2006. 9, 10, 23
- , “A data-driven spoken language understanding system,” in *Workshop on Automatic Speech Recognition and Understanding (ASRU)*. IEEE, Nov 2003, pp. 583–588. 29
- M. Henderson, B. Thomson, and J. Williams, “The Second Dialog State Tracking Challenge,” in *SIGdial*. Association for Computational Linguistics, 2014. iii, 11, 30, 68, 73, 74, 100, 115
- M. Henderson, B. Thomson, and S. Young, “Robust Dialog State Tracking Using Delexicalised Recurrent Neural Networks and Unsupervised Adaptation,” in *Spoken Language Technology Workshop*. IEEE, 2014. iii, 78
- M. Henderson, M. Gašić, B. Thomson, P. Tsiakoulis, K. Yu, and S. Young, “Discriminative Spoken Language Understanding Using Word Confusion Networks,” in *Spoken Language Technology Workshop*. IEEE, 2012. iii, 23, 40, 74
- M. Henderson, B. Thomson, and S. Young, “Deep Neural Network Approach for the Dialog State Tracking Challenge,” in *SIGdial*. Association for Computational Linguistics, 2013. iii, 12, 51, 52
- M. Henderson, B. Thomson, and J. D. Williams, “The Third Dialog State Tracking Challenge,” in *Spoken Language Technology Workshop*. IEEE, 2014. iii, 11, 44, 68, 77, 115
- M. Henderson, B. Thomson, and S. Young, “Word-Based Dialog State Tracking with Recurrent Neural Networks,” in *SIGdial*. Association for Computational Linguistics, 2014. iii, 53, 75
- G. Hinton, L. Deng, D. Yu, G. Dahl, A. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. Sainath, and B. Kingsbury, “Deep neural networks for acoustic modeling in speech recognition,” *Signal Processing Magazine*, 2012. 6, 129
- G. E. Hinton and R. R. Salakhutdinov, “Reducing the dimensionality of data with neural networks,” *Science*, vol. 313, no. 5786, pp. 504–507, 2006. 60
- S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural Comput.*, vol. 9, no. 8, Nov. 1997. 66, 112, 132
- J. J. Hopfield, “Neural networks and physical systems with emergent collective computational abilities,” *Proceedings of the National Academy of Sciences of the United States of America*, vol. 79, no. 8, pp. 2554–2558, Apr. 1982. 129
- E. Horvitz and T. Paek, “A computational architecture for conversation,” in *Proceedings of the Seventh International Conference on User Modeling*, ser. UM ’99. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 1999, pp. 201–210. 3

- A. Isard, J. Oberlander, C. Matheson, and I. Androutsopoulos, "Speaking the users' languages," *IEEE INTELLIGENT SYSTEMS MAGAZINE*, vol. 18, p. 2003, 2003. 15
- M. Jeong and G. Geunbae Lee, "Triangular-chain conditional random fields," *Audio, Speech, and Language Processing, IEEE Transactions on*, vol. 16, no. 7, pp. 1287–1302, Sept 2008. 11, 20, 21, 22
- M. I. Jordan, "Serial order: A parallel distributed processing approach." Institute for Cognitive Science, University of California, Tech. Rep. ICS Report 8604, 1986. 132
- D. Jurafsky and J. H. Martin, *Speech and Language Processing*, 2nd ed. Prentice Hall, 2008. 5, 7
- F. Jurcicek, B. Thomson, and S. Young, "Natural actor and belief critic: Reinforcement algorithm for learning parameters of dialogue systems modelled as POMDPs." *TSLP*, vol. 7, 2011. 71
- R. Kadlec, J. Libovický, J. Macek, and J. Kleindienst, "IBM's belief tracker: Results on dialog state tracking challenge datasets," *EACL 2014*, p. 10, 2014. 12
- R. Kadlec, M. Vodolan, J. Libovický, J. Macek, and J. Kleindienst, "Knowledge-based dialog state tracking," in *Spoken Language Technology Workshop*. IEEE, 2014. 12, 78, 81, 97
- N. Kambhatla, "Combining lexical, syntactic, and semantic features with maximum entropy models for extracting relations," in *Proceedings of the ACL 2004 on Interactive Poster and Demonstration Sessions*, ser. ACLdemo '04. Stroudsburg, PA, USA: Association for Computational Linguistics, 2004. 112
- R. J. Kate and R. J. Mooney, "Using string-kernels for learning semantic parsers," in *Proceedings of ACL*, 2006. 10
- R. J. Kate, Y. W. Wong, and R. J. Mooney, "Learning to transform natural to formal languages," in *Proceedings of AAAI-05*, 2005. 10
- D. Kim, C. J. Choi, K.-E. Kim, J. Lee, and J. Sohn, "Engineering Statistical Dialog State Trackers: A Case Study on DSTC," in *SIGdial*. Association for Computational Linguistics, 2013. 12
- D. Kim, M. Henderson, M. Gašić, P. Tsiakoulis, and S. Young, "The use of discriminative belief tracking in POMDP-based dialogue systems," in *Spoken Language Technology Workshop*. IEEE, 2014. 81
- S. Kim and R. E. Banchs, "Sequential labeling for tracking dynamic dialog states," in *SIGdial*. Philadelphia, PA, U.S.A.: Association for Computational Linguistics, June 2014. 13, 53, 75
- R. Kurzweil, *The Singularity Is Near: When Humans Transcend Biology*. Penguin, 2006.

References

- T. Kwiatkowski, L. Zettlemoyer, S. Goldwater, and M. Steedman, “Lexical generalization in ccg grammar induction for semantic parsing,” in *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, ser. EMNLP ’11. Stroudsburg, PA, USA: Association for Computational Linguistics, 2011, pp. 1512–1523. 9
- J. D. Lafferty, A. McCallum, and F. C. N. Pereira, “Conditional random fields: Probabilistic models for segmenting and labeling sequence data,” in *Proceedings of the Eighteenth International Conference on Machine Learning*, ser. ICML ’01. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2001, pp. 282–289. 11, 20
- S. Larsson and D. R. Traum, “Information state and dialogue management in the TRINDI dialogue move engine toolkit,” *Natural Language Engineering*, vol. 6, no. 3&4, pp. 323–340, Sep. 2000. 13
- B.-J. Lee, W. Lim, and K.-E. Kim, “Optimizing generative dialog state tracker via cascading gradient descent,” in *SIGdial*. Association for Computational Linguistics, 2014, p. 273–281. 12, 75
- S. Lee, “Structured Discriminative Model For Dialog State Tracking,” in *SIGdial*. Association for Computational Linguistics, 2013. 13, 53
- S. Lee and M. Eskenazi, “Recipe For Building Robust Spoken Dialog State Trackers: Dialog State Tracking Challenge System Description,” in *SIGdial*. Association for Computational Linguistics, 2013. 12, 51
- E. Levin and R. Pieraccini, “Chronus, the next generation,” in *Proceedings of the ARPA Workshop on Spoken Language Technology*, 1995. 10
- , “A stochastic model of computer-human interaction for learning dialogue strategies,” in *In EUROSPEECH 97*, 1997, pp. 1883–1886. 13
- Q. Li, G. Tur, D. Hakkani-Tur, X. Li, T. Paek, A. Gunawardana, and C. Quirk, “Distributed open-domain conversational understanding framework with domain-independent extractors,” in *Spoken Language Technology Workshop*. IEEE, December 2014. 20
- B. Lucas, “Voicexml,” *Communications of the ACM*, vol. 43, no. 9, p. 53, 2000. 13
- F. Mairesse, M. Gašić, F. Jurcicek, S. Keizer, B. Thomson, K. Yu, and S. Young, “Spoken language understanding from unaligned data using discriminative classification models,” in *International Conference on Acoustics, Speech, and Signal Processing*. IEEE, 2009. 10, 11, 22, 23, 40
- F. Mairesse and S. Young, “Stochastic language generation in dialogue using factored language models,” in *Computational Linguistics*, 2014, pp. 763–799. 15
- L. Mangu, E. Brill, and A. Stolcke, “Finding consensus among words: lattice-based word error minimisation,” *Computer Speech & Language*, pp. 373–400, 2000. 8, 32
- W. S. McCulloch and W. H. Pitts, “A logical calculus of the ideas immanent in nervous activity,” *Bulletin of Mathematical Biophysics*, vol. 5, pp. 115–133, 1943. 129

- M. McTear, "Modelling spoken dialogues with state transition diagrams: Experiences with the CSLU toolkit," in *ICSLP*, vol. 4, Sydney, 1998. 13
- G. Mesnil, X. He, L. Deng, and Y. Bengio, "Investigation of recurrent-neural-network architectures and learning methods for spoken language understanding," in *Annual Conference of the International Speech Communication Association (INTERSPEECH)*, August 2013. 112
- A. Metallinou, D. Bohus, and J. Williams, "Discriminative state tracking for spoken dialog systems," in *ACL*. The Association for Computer Linguistics, 2013. 12, 51
- I. V. Meza-Ruiz, S. Riedel, and O. Lemon, "Accurate statistical spoken language understanding from limited development resources," in *International Conference on Acoustics, Speech, and Signal Processing*. IEEE, 2008. 10
- S. Miller, D. Stallard, R. Bobrow, and R. Schwartz, "A fully statistical approach to natural language interfaces," in *Proceedings of the 34th Annual Meeting on Association for Computational Linguistics*, ser. ACL '96. Stroudsburg, PA, USA: Association for Computational Linguistics, 1996, pp. 55–61. 9, 10
- J. Moore, M. Ellen, F. Oliver, and L. M. White, "Generating tailored, comparative descriptions in spoken dialogue," in *17th International Florida Artificial Intelligence Research Society Conference*, 2004, pp. 917–922. 15
- M. C. Mozer, "A focused backpropagation algorithm for temporal pattern recognition," Depts. of Psychology and Computer Science, University of Toronto, Toronto, Tech. Rep. CRG-TR-88-3, Jun. 1988. 132
- H. Murveit, J. Butzberger, V. Digalakis, and M. Weintraub, "Large-vocabulary dictation using SRI's DECIPHER™ speech recognition system: progressive search techniques," in *International Conference on Acoustics, Speech, and Signal Processing*. Washington, DC, USA: IEEE, 1993, pp. 319–322. 6
- L.-M. Nguyen, A. Shimazu, and X.-H. Phan, "Semantic parsing with structured SVM ensemble classification models," in *Proceedings of the COLING/ACL on Main Conference Poster Sessions*, ser. COLING-ACL '06. Stroudsburg, PA, USA: Association for Computational Linguistics, 2006, pp. 619–626. 9
- L. M. Norton, D. A. Dahl, D. P. McKay, L. Hirschman, M. C. Linebarger, D. Magerman, and C. N. Ball, "Management and evaluation of interactive dialog in the air travel domain," DTIC Document, Tech. Rep., 1990. 1
- S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on Knowledge and Data Engineering*, 2010. 91
- R. Pieraccini and J. Huerta, "Where do we go from here? research and commercial spoken dialog systems," in *SIGdial*, 2005. 5
- J. Platt, "Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods," *Advances in large margin classifiers*, vol. 10, no. 3, pp. 61–74, 1999. 22

References

- S. Pradhan, W. Ward, K. Hacioglu, J. Martin, and D. Jurafsky, “Shallow semantic parsing using support vector machines,” in *Proceedings of HLT/NAACL*, 2004, p. 233. 10
- G. N. Ramaswamy and J. Kleindienst, “Hierarchical feature-based translation for scalable natural language understanding,” in *Annual Conference of the International Speech Communication Association (INTERSPEECH)*, 2000. 10
- W. J. Rapaport, “Logical foundations for belief representation,” *Cognitive Science*, vol. 10, no. 4, pp. 371–422, 1986. 11
- C. Raymond and G. Riccardi, “Generative and discriminative algorithms for spoken language understanding,” in *Annual Conference of the International Speech Communication Association (INTERSPEECH)*. ISCA, 2007, pp. 1605–1608. 20
- E. Reiter and R. Dale, *Building Natural Language Generation Systems*. New York, NY, USA: Cambridge University Press, 2000. 15
- H. Ren, W. Xu, and Y. Yan, “Markovian discriminative modeling for cross-domain dialog state tracking,” in *Spoken Language Technology Workshop*. IEEE, 2014. 12, 51, 77, 78
- , “Markovian discriminative modeling for dialog state tracking,” in *SIGdial*. Philadelphia, PA, U.S.A.: Association for Computational Linguistics, June 2014. 12, 51
- V. Rieser and O. Lemon, *Reinforcement Learning for Adaptive Dialogue Systems: A Data-driven Methodology for Dialogue Management and Natural Language Generation*. Springer, 2011. 15
- , “Natural language generation as planning under uncertainty for spoken dialogue systems,” in *Empirical methods in natural language generation*. Springer, 2010, pp. 105–120. 15
- T. Robinson, M. Hochberg, and S. Renals, “The use of recurrent neural networks in continuous speech recognition,” in *Automatic Speech and Speaker Recognition*, ser. The Kluwer International Series in Engineering and Computer Science, C. H. Lee, K. K. Paliwal, and F. K. Soong, Eds. Springer US, 1996, vol. 355, pp. 233–258. 6
- N. Roy, J. Pineau, and S. Thrun, “Spoken dialogue management using probabilistic reasoning,” in *ACL*, 2000. 14
- M. Saraclar, “Lattice-based search for spoken utterance retrieval,” in *In Proceedings of HLT-NAACL 2004*, 2004, pp. 129–136. 29
- R. Schapire, M. Rochery, M. Rahim, and N. Gupta, “Boosting with prior knowledge for call classification,” *Speech and Audio Processing, IEEE Transactions on*, pp. 174 – 181, 2005. 9
- R. Schwartz, S. Miller, D. Stallard, and J. Makhoul, “Language understanding using hidden understanding models,” in *ICSLP*, 1996. 10
- J. R. Searle, *Speech Acts: An Essay in the Philosophy of Language*. Cambridge University Press, 1970. 8

- B. Settles, “Active learning literature survey,” University of Wisconsin–Madison, Computer Sciences Technical Report 1648, 2009. 98, 111
- S. Singh, M. Kearns, D. Litman, and M. Walker, “Reinforcement learning for spoken dialogue systems,” 1999. 13
- R. Smith, “Comparative Error Analysis of Dialog State Tracking,” in *SIGdial*. Association for Computational Linguistics, 2014. 12, 75
- M. Steedman, *The Syntactic Process*. Cambridge, MA, USA: MIT Press, 2000. 9
- A. Stent, R. Prasad, and M. Walker, “Trainable sentence planning for complex information presentations in spoken dialog systems,” in *ACL*, Barcelona, Spain, July 2004. 15
- K. Sun, L. Chen, S. Zhu, and K. Yu, “The SJTU system for dialog state tracking challenge 2,” in *SIGdial*. Philadelphia, PA, U.S.A.: Association for Computational Linguistics, June 2014. 12, 30, 45, 51, 75
- , “A generalized rule based tracker for dialogue state tracking,” in *Spoken Language Technology Workshop*. IEEE, 2014. 12, 78
- J. Sundman, “Artificial stupidity,” http://www.salon.com/2003/02/26/loebner_part_one/, 2003. 1
- S. Sutton, D. Novick, R. Cole, P. Vermeulen, J. de Villiers, J. Schalkwyk, and M. Fanty, “Building 10,000 spoken dialogue systems,” in *Spoken Language, 1996. ICSLP 96. Proceedings., Fourth International Conference on*, vol. 2, Oct 1996. 13
- P. Taylor, A. W. Black, and R. Caley, “The architecture of the festival speech synthesis system,” in *ESCA Workshop in speech synthesis*, 1998, pp. 147–151. 15
- B. Thomson and S. Young, “Bayesian update of dialogue state: A POMDP framework for spoken dialogue systems,” *Computer Speech & Language*, vol. 24, no. 4, Oct. 2010. 12, 14, 38, 47, 48, 84
- B. Thomson, K. Yu, M. Gašić, S. Keizer, F. Mairesse, J. Schatzmann, and S. Young, “Evaluating semantic-level confidence scores with multiple hypotheses,” in *Annual Conference of the International Speech Communication Association (INTERSPEECH)*, 2008. 25
- B. Thomson, F. Jurcicek, M. Gasic, S. Keizer, F. Mairesse, K. Yu, and S. Young, “Parameter learning for pomdp spoken dialogue models,” in *Spoken Language Technology Workshop*. IEEE, Dec 2010, pp. 271–276. 50
- B. Thomson, F. Jurcicek, and G. M., “Parameter learning for POMDP spoken dialogue models,” *Spoken Language Technology Workshop*, 2010. 12, 91
- B. Thomson, M. Gašić, M. Henderson, P. Tsiakoulis, and S. Young, “N-best error simulation for training spoken dialogue systems,” in *Spoken Language Technology Workshop*. IEEE, Dec 2012, pp. 37–42. 38
- B. Thomson, “Statistical methods for spoken dialogue management,” Ph.D. dissertation, University of Cambridge, 2009. 13, 29, 47, 48, 106

References

- D. R. Traum, “20 questions on dialogue act taxonomies,” *JOURNAL OF SEMANTICS*, vol. 17, pp. 7–30, 2000. 117
- P. Tsiakoulis, M. Gašić, M. Henderson, J. Prombonas, B. Thomson, K. Yu, S. Young, and E. Tzirkel, “Statistical methods for building robust spoken dialogue systems in an automobile,” in *4th International Conference on Applied Human Factors and Ergonomics*, 2012. 24
- P. Tsiakoulis, C. Breslin, M. Gasic, M. Henderson, D. Kim, M. Szummer, B. Thomson, and S. Young, “Dialogue context sensitive HMM-based speech synthesis,” in *International Conference on Acoustics, Speech, and Signal Processing*. IEEE, May 2014. 15
- G. Tur, D. Hakkani-Tur, and G. Riccardi, “Extending boosting for call classification using word confusion networks,” in *International Conference on Acoustics, Speech, and Signal Processing*, vol. 1. IEEE, May 2004, pp. I-437–40 vol.1. 29
- G. Tur, J. Wright, A. Gorin, G. Riccardi, and D. Hakkani-tür, “Improving spoken language understanding using word confusion networks,” in *ICSLP*, 2002, pp. 1137–1140. 29
- G. Tur, R. E. Schapire, and D. Hakkani-Tür, “Active learning for spoken language understanding,” in *International Conference on Acoustics, Speech, and Signal Processing*. IEEE, 2003. 98
- G. Tur, A. Deoras, and D. Hakkani-Tür, “Semantic parsing using word confusion networks with conditional random fields,” in *Annual Conference of the International Speech Communication Association (INTERSPEECH)*, 2013. 11, 28, 30, 74
- A. M. Turing, R. Braithwaite, G. Jefferson, and M. H. A. Newman, “Can automatic calculating machines be said to think?” in *The Essential Turing*. Oxford University Press (reprinted 2004), 1952, pp. 487–506. 1
- P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol, “Extracting and composing robust features with denoising autoencoders,” in *International Conference on Machine Learning*, ser. ICML ’08. New York, NY, USA: ACM, 2008. 131
- M. A. Walker, O. Rambow, and M. Rogati, “Spot: A trainable sentence planner,” in *NAACL*, Stroudsburg, PA, USA, 2001. 15
- Y.-Y. Wang and A. Acero, “Discriminative models for spoken language understanding,” in *Annual Conference of the International Speech Communication Association (INTERSPEECH)*, 2006. 10, 11, 20
- Y.-Y. Wang, L. Deng, and A. Acero, “Spoken language understanding — an introduction to the statistical framework,” *IEEE Signal Processing Magazine*, vol. 22, no. 5, pp. 16–31, 2005. 18, 20
- Y.-Y. Wang, J. Lee, M. Mahajan, and A. Acero, “Statistical spoken language understanding: from generative model to conditional model,” in *NIPS Workshop: Advances in Structured Learning for Text and Speech Processing*, Whistler, BC, Canada, 2005. 20

- Z. Wang and O. Lemon, "A simple and generic belief tracking mechanism for the dialog state tracking challenge: On the believability of observed information," in *SIGdial*. Association for Computational Linguistics, 2013. 12, 47, 75, 78
- W. Ward, "Extracting Information From Spontaneous Speech," in *Annual Conference of the International Speech Communication Association (INTERSPEECH)*, 1994. 9, 18, 124
- J. Williams and S. Young, "Scaling up POMDPs for Dialog Management: The "Summary POMDP" Method," in *Workshop on Automatic Speech Recognition and Understanding (ASRU)*. IEEE, 2005, pp. 177–182. 3, 14
- J. Williams, "Incremental partition recombination for efficient tracking of multiple dialog states," in *International Conference on Acoustics, Speech, and Signal Processing*. IEEE, 2010. 12
- , "A critical analysis of two statistical spoken dialog systems in public use," in *Spoken Language Technology Workshop*. IEEE, 2012. 12, 50, 51
- , "Web-style ranking and SLU combination for dialog state tracking," in *SIGdial*. Association for Computational Linguistics, 2014. 12, 30, 51, 75, 76, 80, 100
- , "Multi-domain learning and generalization in dialog state tracking," in *SIGdial*. Association for Computational Linguistics, 2013. 12, 51
- J. Williams, I. Arizmendi, and A. Conkie, "Demonstration of AT&T "Let's Go": A production-grade statistical spoken dialog system," in *Proc Demonstration Session at IEEE Workshop on Spoken Language Technology (SLT), Berkeley, California, USA*, 2010. 29
- J. Williams, A. Raux, D. Ramachadran, and A. Black, "The Dialog State Tracking Challenge," in *SIGdial*. Association for Computational Linguistics, August 2013. 11, 68, 72
- J. Williams, M. Henderson, A. Raux, B. Thomson, A. Black, and D. Ramachandran, "The Dialog State Tracking Challenge Series," *AI Magazine*, vol. 35, no. 4, pp. 121–124, 2014. 42, 67
- P. Xu and R. Sarikaya, "Convolutional neural network based triangular CRF for joint intent detection and slot filling," in *Workshop on Automatic Speech Recognition and Understanding (ASRU)*. IEEE, Dec 2013, pp. 78–83. 11
- K. Yao, G. Zweig, M. yuh Hwang, Y. Shi, and D. Yu, "Recurrent neural networks for language understanding," in *Annual Conference of the International Speech Communication Association (INTERSPEECH)*, 2013. 11, 20
- K. Yao, B. Peng, Y. Zhang, D. Yu, G. Zweig, and Y. Shi, "Spoken language understanding using long short-term memory neural networks," in *Spoken Language Technology Workshop*. IEEE, 2014. 11, 20
- S. Young, "CUED standard dialogue acts," *Report, Cambridge University Engineering Department, 14th October*, 2007. 117

References

- S. Young and C. Proctor, “The design and implementation of dialogue control in voice operated database inquiry systems,” *Computer Speech & Language*, vol. 3, no. 4, pp. 329–353, 1989. 9
- S. Young, G. Evermann, M. Gales, T. Hain, D. Kershaw, G. Moore, J. Odell, D. Ollason, D. Povey, V. Valtchev, and P. Woodland, *The HTK Book, version 3.4*. Cambridge, UK: Cambridge University Engineering Department, 2006. 18
- S. Young, “Talking to machines (statistically speaking),” in *ICSLP*, 2002. 9, 13
- S. Young, M. Gašić, S. Keizer, F. Mairesse, J. Schatzmann, B. Thomson, and K. Yu, “The Hidden Information State model: A practical framework for POMDP-based spoken dialogue management,” *Computer Speech & Language*, vol. 24, no. 2, pp. 150–174, Apr. 2010. 11
- S. Young, C. Breslin, M. Gašić, M. Henderson, D. Kim, M. Szummer, B. Thomson, P. Tsakoulis, and E. Tzirkel Hancock, “Evaluation of Statistical POMDP-based Dialogue Systems in Noisy Environment,” in *Proceedings of IWSDS*, 2013. 69, 83
- S. J. Young, “Probabilistic methods in spoken–dialogue systems,” *Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, vol. 358, no. 1769, pp. 1389–1402, 2000. 3
- J. M. Zelle and R. J. Mooney, “Learning to parse database queries using inductive logic programming,” in *Proceedings of the Thirteenth National Conference on Artificial Intelligence - Volume 2*, ser. AAAI’96. AAAI Press, 1996, pp. 1050–1055. 9
- H. Zen, T. Nose, J. Yamagishi, S. Sako, T. Masuko, A. W. Black, and K. Tokuda, “The HMM-based speech synthesis system (HTS) version 2.0,” in *Sixth ISCA Workshop on Speech Synthesis (SSW6)*. ISCA, 2007, pp. 294–299. 15
- H. Zen, A. Senior, and M. Schuster, “Statistical parametric speech synthesis using deep neural networks,” in *International Conference on Acoustics, Speech, and Signal Processing*, 2013, pp. 7962–7966. 15
- H. Zen, H. Sak, A. Graves, and A. Senior, “Statistical Parametric Speech Synthesis based on Recurrent Neural Networks,” in *UKSpeech*, 2014. 15
- L. Zettlemoyer and M. Collins, “Online learning of relaxed CCG grammars for parsing to logical form,” in *Proceedings of EMNLP-CoNLL*, 2007. 9, 10, 23, 112
- D. Zhou and Y. He, “Learning conditional random fields from unaligned data for natural language understanding,” in *Advances in Information Retrieval*, ser. Lecture Notes in Computer Science. Springer Berlin / Heidelberg, 2011, pp. 283–288. 11
- X. Zhu and A. B. Goldberg, “Introduction to semi-supervised learning,” *Synthesis Lectures on Artificial Intelligence and Machine Learning*, vol. 3, 2009. 91
- V. Zue, S. Seneff, J. Glass, J. Polifroni, C. Pao, T. Hazen, and L. Hetherington, “JUPITER: a telephone-based conversational interface for weather information,” *Speech and Audio Processing, IEEE Transactions on*, vol. 8, no. 1, pp. 85–96, Jan 2000. 9