Recurrent Neural Network Language Generation for Dialogue Systems

Tsung-Hsien Wen

Department of Engineering
University of Cambridge

This dissertation is submitted for the degree of
Doctor of Philosophy

Darwin College

March 2018

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 40277 words including appendices, bibliography, footnotes, tables and equations and has 53 figures and tables. Some of the work presented here was published in the Special Interest Group on Discourse and Dialogue (Wen et al., 2015a), the Empirical Methods on Natural Language Processing (Wen et al., 2016a, 2015c), the North American Chapter of the Association for Computational Linguistics (Wen et al., 2016b), the European Chapter of the Association for Computational Linguistics (Wen et al., 2017c), and the International Conference of Machine Learning (Wen et al., 2017a).

Tsung-Hsien Wen
March 2018

Acknowledgements

First, I would like to thank my supervisor Steve Young for his dedicated supervision throughout the course of my PhD. It has always been a pleasure talking to him, which is where he taught me how to think clearly and perform effective research. His feedback is always well-considered and to the point, but he never stops his research group from exploring interesting research ideas. He has been, in general, an inspiration and it's really my pleasure to be his PhD student. I'm also grateful for my advisor Milica Gasic, for taking me to the group and her help and guidance during the study of the course.

It has been a privilege to work with the Cambridge Dialogue Systems Group, which has provided a highly motivating and friendly environment for research. I would like to thank some of the group's previous members, David Vandyke, Dongho Kim and Pirros Tsiakoulis for their guidance on the group infrastructure and Filip Jurcicek on his excellent work setting up the Amazon Mechanical Turk. I would also like to thank the current group members Stefan Ultes, Lina Maria Rojas Barahona, Inigo Casanueva, and Pawel Budzianowski for their invaluable discussion and support, Anna Langley and Patrick Gosling for their computing assistance, as well as Rachel Fogg and Diane Hazell for their excellent administrative work.

I would like to extend my gratitude to my peers Nikola Mrksic and Pei-Hao Su for their mutual support. It has been a great pleasure to be able to pursue my PhD with them. I would like to particularly acknowledge Pei-Hao Su for his support these past ten years, which we have spent studying and working together. It has been a great journey. Outside of the department, I would like to acknowledge Yishu Miao for his invaluable discussion on latent variable modelling. His cooperation has helped to shape the last part of this thesis.

I acknowledge Toshiba Cambridge Research Laboratory for funding and Yannis Stylianou, Zhuoran Wang, Alex Papangelis, Margarita Kotti, and Norbert Braunschweiler for their valuable comments and suggestions.

Lastly, I would like to offer a sincere thank you to my parents, Wen-Bing and Chiu-Yuan, for their caring encouragement. I couldn't have finished this thesis without their support. I'm also very grateful to my girlfriend, Fang-Yu Chiu, for her companionship throughout this journey.

Abstract

Language is the principal medium for ideas, while dialogue is the most natural and effective way for humans to interact with and access information from machines. Natural language generation (NLG) is a critical component of spoken dialogue and it has a significant impact on usability and perceived quality. Many commonly used NLG systems employ rules and heuristics, which tend to generate inflexible and stylised responses without the natural variation of human language. However, the frequent repetition of identical output forms can quickly make dialogue become tedious for most real-world users. Additionally, these rules and heuristics are not scalable and hence not trivially extensible to other domains or languages. A statistical approach to language generation can learn language decisions directly from data without relying on hand-coded rules or heuristics, which brings scalability and flexibility to NLG. Statistical models also provide an opportunity to learn in-domain human colloquialisms and cross-domain model adaptations.

A robust and quasi-supervised NLG model is proposed in this thesis. The model leverages a Recurrent Neural Network (RNN)-based surface realiser and a gating mechanism applied to input semantics. The model is motivated by the Long-Short Term Memory (LSTM) network. The RNN-based surface realiser and gating mechanism uses a neural network to learn end-to-end language generation decisions from input dialogue act and sentence pairs; it also integrates sentence planning and surface realisation into a single optimisation problem. The single optimisation not only bypasses the costly intermediate linguistic annotations but also generates more natural and human-like responses. Furthermore, a domain adaptation study shows that the proposed model can be readily adapted and extended to new dialogue domains via a proposed recipe.

Continuing the success of end-to-end learning, the second part of the thesis speculates on building an end-to-end dialogue system by framing it as a conditional generation problem. The proposed model encapsulates a belief tracker with a minimal state representation and a generator that takes the dialogue context to produce responses. These features suggest comprehension and fast learning. The proposed model is capable of understanding requests and accomplishing tasks after training on only a few hundred human-human dialogues. A complementary Wizard-of-Oz data collection method is also introduced to facilitate the collection of human-human conversations from online workers. The results demonstrate that the proposed model can talk to human judges naturally, without any difficulty, for a sample application domain. In addition, the results also suggest that the introduction of a stochastic latent variable can help the system model intrinsic variation in communicative intention much better.

Table of contents

List of figures xiii

List of tables xvii

1 Introduction 1

1.1 Thesis Outline and Contributions . . . 3

2 Overview of Spoken Dialogue Systems 7

2.1 Speech Interface . . . 8

2.2 Spoken Language Understanding . . . 9

2.3 Dialogue State Tracking . . . 11

2.4 Dialogue Management . . . 12

2.5 Natural Language Generation . . . 13

2.5.1 Template-based Language Generator . . . 14

2.5.2 Linguistically Motivated Approaches . . . 15

2.6 Conclusion . . . 18

3 Corpus-based Natural Language Generation for Dialogue Systems 19

3.1 Class-based Language Model . . . 19

3.2 Phrase-based Factored Language Model . . . 21

3.3 Example-based Approaches . . . 23

3.4 Attention-based Sequence-to-Sequence Approaches . . . 23

3.5 Evaluation Metrics and Difficulties . . . 25

3.5.1 BLEU score . . . 25

3.5.2 Slot Error Rate . . . 27

3.5.3 Language Variability . . . 28

3.6 Conclusions . . . 28

4 Neural Networks and Deep Learning 29
4.1 Neural Networks 29
4.1.1 Recurrent Neural Networks 32
4.1.2 Long Short-term Memory 33
4.1.3 Bidirectional Networks 36
4.1.4 Convolutional Neural Networks 37
4.2 Objective Function 39
4.2.1 Maximum Likelihood 39
4.2.2 Discriminative Training 40
4.3 Stochastic Neural Networks 41
4.3.1 Conditional Variational Autoencoder Framework 41
4.3.2 Neural Variational Inference 42
4.4 Conclusion 45

5 Recurrent Neural Network Language Generators 47
5.1 The Recurrent Language Generation Framework 47
5.2 Gating Mechanism 49
5.2.1 The Heuristically Gated LSTM Generator 49
5.2.2 The Semantically Controlled LSTM Generator 50
5.3 Attention Mechanism 52
5.3.1 The Attentive Encoder Decoder LSTM Generator 52
5.4 Data Collection 54
5.4.1 Ontologies 54
5.4.2 Methodology 55
5.4.3 Dataset Overview 55
5.5 Baselines 56
5.6 Corpus-based Evaluation 57
5.6.1 Experimental Setup and Evaluation Metrics 57
5.6.2 Results 58
5.7 Human Evaluation 61
5.7.1 Experimental Setup and Evaluation Metrics 61
5.7.2 Results 62
5.8 Conclusions 64

6 Domain Adaptation 65
6.1 Data Augmentation 65
6.2 Feature-based Domain Adaptation 66

6.3 Model-based Domain Adaptation 68
6.4 Baseline Method - Model Fine-Tuning 69
6.5 The Adaptation Recipe for Language Generators 69
6.5.1 Data Counterfeiting 70
6.5.2 Discriminative Training for Adaptation 71
6.6 Corpus-based Evaluation 72
6.6.1 Experimental Setup and Evaluation Metrics 72
6.6.2 Results 73
6.7 Human Evaluation 76
6.7.1 Experimental Setup and Evaluation Metrics 76
6.7.2 Results 76
6.8 Conclusions 78

7 Generation by Conditioning on a Broader Context 79
7.1 End-to-End Dialogue Modelling 80
7.2 Goal-oriented Dialogue as Conditional Generation 82
7.3 Neural Dialogue Model 83
7.3.1 Intent Network 84
7.3.2 The RNN-CNN Dialogue State Tracker 86
7.3.3 Database Operator and Deterministic Policy Network 88
7.3.4 Conditional LSTM Generator and Its Variants 89
7.4 Wizard-of-Oz Data Collection 92
7.5 Evaluation on Neural Dialogue Models 93
7.5.1 Experimental Setup and Evaluation Metrics 93
7.5.2 Results 95
7.5.3 Human Evaluation 96
7.6 Conclusions 97

8 Generation based on a Latent Policy 101
8.1 Stochastic Policy as a Discrete Latent Variable 102
8.2 Latent Intention Dialogue Model 103
8.2.1 Model 103
8.2.2 Inference 105
8.2.3 Semi-supervised Learning 107
8.2.4 Reinforcement Learning 108
8.3 Evaluation on Latent Intention Dialogue Models 109
8.3.1 Experimental Setup 109

8.3.2 Results 110
8.3.3 Human Evaluation 112
8.4 Conclusions 114

9 Conclusions 117
9.1 Limitations and Future Work 118

Appendix A Slot-based Dialogue Domains 121

Appendix B Dialogue Act Format 123

Appendix C Example of a Template-based Generator 127

Appendix D The Wizard-of-Oz website 131

References 137

List of figures

2.1 Dialogue system architecture 7

3.1 An example of the sentence for the "*inform*" DA and its corresponding stack representation and tree equivalent used in Mairesse and Young (2014). Figure borrowed from Mairesse and Young (2014). 21

3.2 An overview of the phrase-based DBN model for NLG. The lower level is a semantic DBN consisting of both mandatory (blue) and functional stacks (white), while the upper level is a phrase-based DBN that oversees mapping semantic stacks to phrases. 22

3.3 Examples adopted from Dušek and Jurcicek (2016). Trees encoded as sequences used for the seq2seq generator (top) and the reranker (bottom). . 24

3.4 Figure adopted from Dušek and Jurcicek (2016). The proposed attention-based seq2seq generator architecture. 24

3.5 Figure adopted from Dušek and Jurcicek (2016). The proposed reranker architecture. 25

4.1 A feed-forward Artificial Neural Network 30

4.2 Architecture of the Elman recurrent neural network. The recurrence is fully connected between two hidden layers in adjacent time steps. 33

4.3 An unfolded view of RNN. Each rectangle represents a layer of hidden units in a single time step. The weighted connections from input to hidden layer are $\mathbf{W}_{\mathrm{x}}$, those from hidden to output are $\mathbf{W}_{\mathrm{h}}$, and the hidden to hidden weights are $\mathbf{W}_{\mathrm{r}}$. Note that the same weights are reused at every time step. . 34

4.4 In the RNN vanishing gradient problem, the colour of nodes is determined by the sensitivity of a single node to input at time 0. Due to the repeated multiplication of the recurrent weights, the influence of the input will vanish or grow over time. The network will eventually forget the first input. 35

4.5 A Long short-term memory cell is an RNN architecture with the memory block $\mathbf{C}_t$. $\mathbf{i}_t$, $\mathbf{f}_t$,$\mathbf{o}_t$ are input, forget, and output gates. $\mathbf{x}_t$ and $\mathbf{h}_{t-1}$ are controlling signals of the three gates. Typically $\mathbf{x}_t$ is the current time step's input and $\mathbf{h}_{t-1}$ is the hidden layer's representation at previous time step. 36

4.6 The unfolded bidirectional RNN. The backward layer and forward layer can encode information from both directions. Six sets of distinct weights are included. Note there are no connections between two hidden layers. 37

4.7 Architecture of a CNN model called LeNet-5 (Lecun et al., 1998a). Each plane is a feature map to capture certain patterns of the previous layer. A fully connected FNN is attached on top as the classifier for recognising input digits. 38

4.8 Gradient estimation in stochastic computation graphs. Figure adapted from "Categorical Re-parameterization with Gumbel-Softmax" by Jang et al. (2016). (a) the gradient $\partial f(h)/\partial h$ can be directly computed via back-propagation if h is deterministic and differentiable. (b) The introduction of a stochastic node z precludes back-propagation because the sampling function $z^{(n)} \sim p_\theta(z|x)$ doesn't have a well-defined gradient. (c) The re-parameterisation trick (Kingma and Welling, 2014; Rezende et al., 2014), which creates an alternative gradient path to circumvent the sampling function inside the network, allows the gradient to flow from $f(z)$ to θ. Here we used the Gaussian re-parameterisation trick as an example, but other types of distributions can also be applied. (d) The scoring function-based gradient-estimator obtains an unbiased estimate of $\nabla f(z)$ by back-propagating along a surrogate loss $\hat{f}\nabla \log p_\theta(z|x)$, where $\hat{f} = f(z) - b(z)$ and $b(z)$ is the baseline for variance reduction. 43

5.1 The RNNLG framework for language generation. 48

5.2 The Heuristically Gated LSTM generator (H-LSTM). At each time step an LSTM cell is used to process the input token and previous time step hidden state. The DA gate is controlled by matching the current generated token with a predefined semantic dictionary. 50

5.3 The Semantically Conditioned LSTM generator (SC-LSTM). The upper part is a traditional LSTM cell in charge of surface realisation, while the lower part is a sentence planning cell based on a sigmoid control gate and a DA. . 51

5.4 The Attention-based Encoder Decoder generator (ENC-DEC). The DA vector is created by concatenating a deterministic DA embedding with an attentive slot-value embedding over potential slot-value pairs. The dashed lines show how the attentive weights are created by the two control signals. 53
5.5 Examples showing how the SC-LSTM controls the dialogue features, flowing into the network through the reading gates. Despite errors due to sparse training data for some slots, each gate generally learned to detect the words and phrases relating to each associated slot-value pair. 60

6.1 An example of the data counterfeiting approach. Note the slots highlighted *red* are randomly sampled from the target domain where the prerequisite is that the counterfeited slot needs to be in the same functional class as the original slot. 71
6.2 Hotel to Restaurant domain adaptation 73
6.3 Restaurant to Hotel domain adaptation 73
6.4 TV to Laptop domain adaptation 74
6.5 Laptop to TV domain adaptation 74
6.6 Restaurant + Hotel to Laptop + TV domain adaptation 75
6.7 Laptop + TV to Restaurant + Hotel domain adaptation 75

7.1 The proposed Neural Dialogue Model framework 84
7.2 Tied Jordan-type RNN belief tracker with delexicalised CNN feature extractor. The output of the CNN feature extractor is a concatenation of top-level sentence (green) embedding and several levels of intermediate n-gram-like embedding's (red and blue). However, if a value cannot be delexicalised in the input, its ngram-like embedding's will be padded with zeros. We zero-pad vectors (in gray) before each convolution operation to make sure the representation at each layer has the same length. The output of each tracker $\mathbf{b}_s^t$ is a distribution over values of a particular slot s. 85
7.3 Three different conditional generation architectures. 89
7.4 The action vector embedding $\mathbf{z}_t$ generated by the vanilla NDM model. Each cluster is labelled with the first three words that the embedding generated. . 96

8.1 LIDM for Goal-oriented Dialogue Modelling 104

D.1 The user webpage. The worker who plays a user is given a task to follow. For each mturk HIT, he/she needs to type an appropriate sentence to carry on the dialogue by looking at both the task description and the dialogue history. 131

D.2 The wizard page. The wizard's job is slightly more complex: the worker needs to go through the dialogue history, fill in the form (top green) by interpreting the user input at this turn, and type in an appropriate response based on the history and the DB result (bottom green). The DB search result is updated when the form is submitted. The form can be divided into informable slots (top) and requestable slots (bottom), which contain all the labels we need to train the trackers. 131

List of tables

5.1 Ontologies of the Restaurant, Hotel, Laptop and TV domains 54

5.2 Properties of the four datasets collected 56

5.3 Corpus-based evaluation on four domains. The results were produced by training each model on 5 random seeds (1-5) and selecting the models with the best BLEU score on the validation set. Results of both the test and the validation set are reported. The approach with the best performance on a metric in a domain is highlighted in **bold**, while the worst is underlined. . . 59

5.4 Samples of top 5 realisations from the SC-LSTM output. 61

5.5 Human evaluation of the Restaurant domain. The significance of the *Utterance Quality Evaluation* section indicates whether a model is *significantly worse* than the reference based on a Wilcoxon signed-rank test. The *Pairwise Preference Comparison* section indicates whether a model is *significantly preferred* to its opponent based on a two-tailed binomial test. 62

5.6 Human evaluation of the Laptop domain. The significance flag in the *Utterance Quality Evaluation* section indicates whether a model is *significantly worse* than the reference based on a Wilcoxon signed-rank test, while in the *Pairwise Preference Comparison* section it indicates whether a model is *significantly preferred* to its opponent based on a two-tailed binomial test. . 63

6.1 Human evaluation of utterance quality in two adaption scenarios. Results are shown for two metrics (rated out of 3 possibilities). Statistical significance was computed using a Wilcoxon signed-rank test, between the model trained with full data (*scrALL*) and all others. 76

6.2 Pairwise preference test among four approaches in two adaption scenarios. Statistical significance was computed using a two-tailed binomial test. . . . 77

7.1 Corpus-based experiment by comparing different NDM architectures. The results were obtained by training models with several hyper-parameter settings. The testing performance is based on the best model of the validation set. 95
7.2 Human assessment of NDM. The rating for comprehension and naturalness are both out of 5. 95
7.3 A comparison of the NDM with a rule-based modular system (*HDC*). A two-tailed binomial test was used. 95
7.4 Samples of real text conversations between online judges and NDM. 99

8.1 An example of the automatically labelled response seed set for semi-supervised learning during variational inference. 110
8.2 Corpus-based experiment by comparing different NDM and LIDM architectures. The results were obtained by training models with several hyper-parameter settings and report the testing performance based on the best model on the validation set. The best performance of each metric in each big block is highlighted in **bold**. 111
8.3 Human evaluation based on text-based conversations. 112
8.4 A sample dialogue from the LIDM, I=100 model, one exchange per block. Each induced latent intention is shown using an (index, probability) tuple, followed by a decoded response. The sample dialogue was produced by following the responses highlighted in bold. 113
8.5 Two sample dialogues from the LIDM+RL, I=100 model, one exchange per block. Comparing to Table 8.4, the RL agent demonstrates a much greedier behavior toward task success. This can be seen in *block 2 & block 4* in which the agent provides the address and phone number even before the user asks. 114

A.1 Slots in the restaurant, hotel, laptop, and TV domains respectively. All the informable slots are also requestable. The group of $\mathrm{S_{req}} \backslash \mathrm{S_{inf}}$ shows the requestable slots that are not informable. Note the type slot contains only one value, which is the domain string. 122

B.1 The set of dialogue acts used in RNNLG. Note that the dialogue act definition here is different from Young (2007) in which the *inform* act is broken down into specific categories to facilitate data collection and learning. 125

C.1 A sample snapshot of the root rules of the template-based generator used in this thesis. Note that both slot and value can be variables. The matching is done by matching as many slot-value pairs as possible. 128
C.2 A sample snapshot of the lower level rules. 129

Chapter 1

Introduction

The development of conversational skills is an important part of human socialisation. Dialogues (or conversations) between people facilitate the exchange of ideas, experiences and knowledge. If we consider the entire world wide web (WWW) as a database that stores known facts, then the most natural interface for accessing this knowledge would be through direct conversations with an Artificial Intelligence agent that can access the web. This thesis focuses on developing dialogue systems, which are the prototypes for the AI agents mentioned above.

The AI field is based on the Dartmouth proposal (McCarthy et al., 1955), which claims that human intelligence "*can in principle be so precisely described that a machine can be made to simulate it*". Turing's imitation game (Turing, 1950) was designed to test this hypothesis by asking human participants to judge whether they are talking to a human or a computer system. The Loebner Prize (Epstein, 1992), a derivative of the Turing test for text-based dialogue systems, has not received a prominent success yet since its inception almost 26 years ago.

An early effort that appears to pass the Turing test is ELIZA (Weizenbaum, 1966), a chatting program developed by Joseph Weizenbaum. ELIZA works by examining keywords in the user's request. If a keyword is found, the resulting response is formed by transforming the user input based on a handcrafted rule. On the other hand, if a keyword is not found, ELIZA replies either with a clever riposte or one of the earlier responses. Although ELIZA could fool some people into believing that they were talking to a human, it was not considered AI because it knows nothing about the real world and only acts upon a set of pre-programmed scripts.

Although open-domain, AI-based dialogue systems that can handle arbitrary conversations with encyclopaedic knowledge are appealing, developing such systems has proven challenging (Lyons, 2007) and the current state-of-the-art is still very far away. Therefore, this thesis focuses on building goal-oriented dialogue systems for applications whose domains are much more constrained, so that the system can assist users with well-defined tasks, such as tourist guidance (Misu and Kawahara, 2007), movie searches (Liu et al., 2012), flight reservations (Seneff and Polifroni, 2000), and information retrieval (Wen et al., 2013b, 2012b).

Due to the complexity of developing end-to-end dialogue systems, state-of-the-art approaches divide the problem into three components: language understanding, dialogue management, and language generation. This modular system design relies on the dialogue act protocol (Traum, 1999) to represent the semantics and communication between system modules. In the past few decades, research on goal-oriented dialogue systems has been focused on either building better language comprehension engines (Allen, 1995; Thomson and Young, 2010), so that the system can understand human queries better, or developing better dialogue managers (Roy et al., 2000a; Young et al., 2010), so that the system can help users accomplish tasks efficiently. Language generation was largely overlooked and handcrafted templates became common practise in the dialogue system development process (Cheyer and Guzzoni, 2007; Mirkovic and Cavedon, 2011).

The Natural Language Generation (NLG) component provides much of the personality of a dialogue agent, which has a significant impact on a user's impression of the system. Dialogue systems with strong comprehension and decision-making capabilities may be effective taskmasters, but they often fail at engaging users in conversations if a rigid, templated-based generator is used to render text. Moreover, critics argue that, through careful engineering efforts and resources, a handcrafted system can still achieve an impressive result. Nevertheless, the handcrafted generator still poses a problem of scaling up its domain coverage. To allow machine responses to be more natural and provide a more engaging conversational flow, a statistical framework trained on data is used in this thesis.

The key in handling complex-structured prediction problems, such as NLG with only lightweight labelling efforts, is the use of Recurrent Neural Network (RNN) language models for sequence generation (Mikolov et al., 2010; Sutskever et al., 2011). An RNN directly learns the mapping from an arbitrary vector representation to surface forms by generating words one-by-one, each from its hidden state. The word embedding (Mikolov et al., 2013), which the model learns during training, grants the model the ability to generalise to rare or unseen examples. Meanwhile, the recurrent connection also empowers the model the potential to explore long-term dependencies from data. The flexible structural design of neural networks also allows the model to quickly adapt to specific problems so that additional features can be easily studied and exploited.

1.1 Thesis Outline and Contributions

The thesis is split into three main sections. After an overview of Spoken Dialogue Systems in Chapter 2, chapters 3, 4, and 5 focus on Natural Language Generation from dialogue act taxonomies, while Chapter 6 studies the domain scalability of the proposed NLG methods. Chapters 7 and 8 then look at the wider context of NLG and frame it as an end-to-end dialogue modelling problem. Below is a description of each of these chapters and their contributions.

Chapter 2, Overview of Spoken Dialogue Systems This chapter presents an overview of a modular Spoken Dialogue System architecture design. A series of system modules are reviewed and the corresponding state-of-the-art approaches are presented. Two classes of Natural Language Generation (NLG) approaches are also introduced: (1) the template-based approaches, and (2) the linguistic-motivated approaches. The aim of this chapter is to lay the foundations of statistical dialogue modelling and NLG and set-up the context of the thesis.

Chapter 3, Corpus-based Natural Language Generation for Dialogue Systems This chapter reviews the corpus-based NLG methods and their applications for dialogue systems, which includes the class-based language model, phrase-based factored language model, and example-based methods for NLG. Evaluation of NLG and its difficulties are also presented. This is where the BLEU score and slot error rate metrics are both introduced. These two metrics are then used as the primary objective metrics for evaluating the NLG models proposed in this thesis.

Chapter 4, Neural Networks and Deep Learning This chapter introduces the basic concept of Neural Network (NN), from the model formulation, optimisation, objective function, to its variants like Recurrent Neural Network (RNN) and Convolutional Neural Network (CNN). Moreover, the second part of the chapter extends the deterministic NN models to stochastic ones and introduces the Conditional Variational Autoencoder (cVAE) framework and its optimisation techniques. The goal of this chapter is to present *building blocks* and optimisation algorithms used in this thesis.

Chapter 5, Recurrent Neural Network Language Generator This chapter extends the idea of Recurrent Neural Network Language Model (Mikolov et al., 2010) and proposes Recurrent Neural Network Language Generation (RNNLG), which integrates an RNN LM as the surface realiser to produce sentences in a sequential way. This generation process is additionally reliant on a dialogue act representation and managed by either a gating mechanism or an attention mechanism to ensure it conveys the intended meaning. Three RNNLG models are proposed, the Heuristically Gated LSTM (H-LSTM) and the Semantically Conditioned LSTM (SC-LSTM), which employ controlling gates and the Attention-based Encoder Decoder (ENC-DEC) with attention. In the experimental section, four corpora are presented. Each of them is collected via Amazon Mechanical Turk and labelled in the form of a dialogue act and sentence pair. The three RNNLG models and baselines are then evaluated. The evaluation is based on a corpus-based evaluation and a human evaluation on the four corpora. Result shows that the SC-LSTM by integrating a learnable gate for latent sentence planning.

Chapter 6, Domain Adaptation This chapter investigates the domain scalability of the proposed SC-LSTM generator. A set of domain adaptation approaches are reviewed and a recipe for adapting the NLG model is proposed based on the survey. The recipe encompasses two methods: data counterfeiting and discriminative training, which are both validated by a corpus-based evaluation and human evaluation. By bootstrapping from data of existing domains, the result demonstrates that the proposed adaptation recipe can produce a very competitive NLG model with only a few hundred in-domain training examples.

Chapter 7, Generation by Conditioning on a Broader Context Speculating on the possibility of training NLG based on a broader dialogue context, this chapter presents the Neural Dialogue Model (NDM) which directly learns end-to-end dialogue decisions from data without the provision of dialogue act annotations. The proposed NDM contains several modularly-connected neural network components which are like that of a Partially Observable Markov Decision Process (POMDP) system. Unlike POMDP-based systems, NDM directly learns dialogue strategies together with language comprehension and generation via supervised learning. The experimental results show that NDM is able to generate appropriate sentences directly from the encoded dialogue context and help the user to accomplish tasks. A series of studies also follow to analyse and understand the representations learnt by NDM.

Chapter 8, Generation based on Latent Policy In this chapter we continue to pursue the end-to-end dialogue modelling problem and extend NDM to the Latent Intention Dialogue Model (LIDM) by introducing a discrete latent variable with the aim of modelling the variations in the dialogue response. The conditional Variational Autoencoder (VAE) framework is employed and proven to be useful to the model, i.e., meaning variation exists in dialogue. In a corpus-based evaluation, LIDM outperforms its deterministic counterpart NDM. Human judges also confirm that LIDM can carry out more natural conversations than NDM. We believe this is a promising step forward, particularly for building autonomous dialogue agents since the learnt discrete latent variable interface not only enables the agent to perform learning using several paradigms but it also demonstrates a competitive performance and a strong scaling potential.

Finally, chapter 9 concludes the thesis, providing a critical analysis of the benefits and limitations of the proposed approaches for NLG. We discuss how these approaches may help to create a real-world dialogue system that is much more natural and scalable, yet with a much shorter development life cycle.

Chapter 2

Overview of Spoken Dialogue Systems

This chapter provides an overview of Spoken Dialogue Systems (SDS) theory. As shown in Figure 2.1, a typical SDS contains five components arranged in a pipeline. There is no common agreement on a single dialogue system architecture in the literature. A representative reference can be found in (Pieraccini and Huerta, 2008). One cycle through the pipeline is a dialogue *turn*, which involves both a user utterance and a system utterance. This thesis focuses on task-oriented dialogue systems, where the task is information-seeking and the domain can be defined by a set of *slots* and their corresponding *values*. Slots are variables that the user can either specify or ask about in the given domain. For example, in a restaurant booking dialogue system, the user may look for restaurants of a particular food type and price range slots and ask for the address and phone number slots. Appendix A provides a more detailed discussion of the slot-based dialogue domains used in this thesis. The subsequent sections look at each of these system-components in detail.

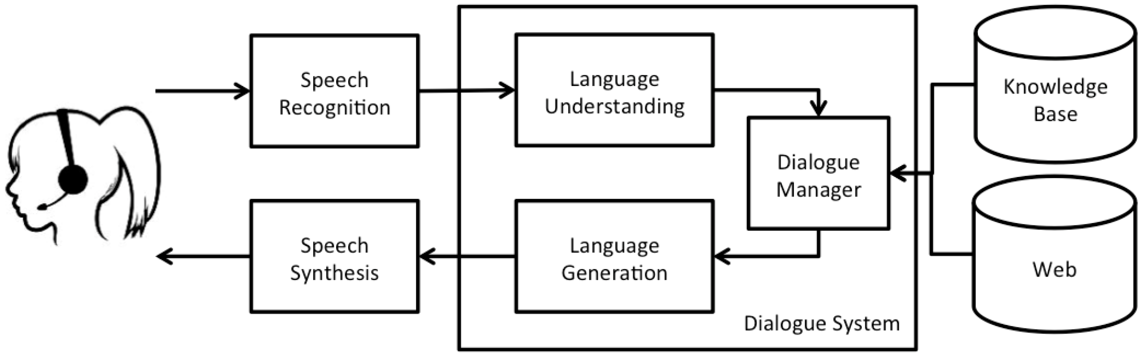


Fig. 2.1 The pipeline architecture of a typical dialogue system.

2.1 Speech Interface

Automatic Speech Recognition (ASR) and Speech Synthesis are components at the two ends of the SDS that provide a speech interface with the user. In modern SDS development, these two components are usually implemented separately from the others; in this way, the speech interface can be generally applicable across a variety of dialogue domains.

Many traditional ASR systems use generative graphical models such Hidden Markov Models (HMMs) to model the acoustics of speech signals (a review is given in Gales and Young (2007)) and n-gram-based language models to capture the transition between word symbols (Goodman, 2001). In recent years, however, this approach has been largely replaced by discriminative artificial neural networks in both acoustic modelling (Deng et al., 2013; Hinton et al., 2012) and language modelling (Bengio et al., 2003; Mikolov et al., 2010). There is even work on doing end-to-end ASR using Recurrent Neural Networks (RNNs) (Amodei et al., 2016; Graves and Jaitly, 2014). Deep neural networks can learn several layers of input representations along with top-level objectives so that we do not need to rely on handcrafted feature extractors like MFCC (Logan, 2000) to represent the raw inputs, but instead model them directly. This additional flexibility renders deep neural networks a very powerful function approximator and the current state-of-the-art classifier used in many machine perception applications.

The output form of an ASR component is particularly important for SDS research because recognition errors impose uncertainty in the dialogue system and decrease its robustness. An ASR component assigns a posterior probability to the syllables of an utterance given its acoustics; thus, a typical form of output is an *N-best list* of hypotheses and their corresponding probabilities. The resulting top hypotheses usually have the same meaning; variations only appear in articles and other short-function words. Other output forms, such as word lattices (Murveit et al., 1993) and word confusion networks (Mangu et al., 2000) are also possible choices. These multiple hypotheses provide a more robust estimation of the user speech making the system more resistant to noise.

The speech synthesis component, on the other hand, takes the system's response and converts it back to speech. The most common approach employed by most systems is based on the unit selection method. In the unit selection method, waveforms are constructed by concatenating segments of recordings held in a database such as the Festival synthesiser (Taylor et al., 1998). Another widely used approach is the HMM-based synthesiser, which is a statistical method using a generative model of speech (Zen et al., 2007). This statistical method can learn dialogue with context-sensitive behaviour if provided with more natural speech (Tsiakoulis et al., 2014). Recently, deep neural networks have also been applied to speech synthesis (Zen et al., 2013). The most famous approach is WaveNet (van den Oord et al., 2016), a fully convolutional neural network (CNN) with various dilation parameters that allow receptive fields to grow exponentially with depth over thousands of time steps. Despite the challenge in runtime computation efficiency, this approach can generate more natural speech and reduce the gap with human performance by over 50%.

2.2 Spoken Language Understanding

Spoken Language Understanding (SLU), the first component of a dialogue system, after the speech interface, identifies and extracts the semantics of an utterance given a user's input text or the output of the ASR. Although many forms of semantic representations are available, most existing spoken dialogue systems use a shallow level of semantics called DA(dialogue act) (Traum, 1999); DA is derived from the concept of the speech act (Searle, 1969). Most DA taxonomies are designed to capture just enough meaning in an utterance to facilitate rational system behaviours within the domain; they are limited by the semantics it can model and therefore constrain the scalability and learnability of the system. A SLU component (or *semantic decoder*) then takes the utterance as an input and maps it to an output DA representing user semantics.

Many spoken dialogue systems use semantic template grammar, such as the Phoenix parser (Ward, 1990), to extract the DA from the utterance (Young and Proctor, 1989; Zue et al., 2000). These approaches first produce a parse tree of the input sentence, which is based on a set of handcrafted rules that map words or phrases to their corresponding semantic concepts. The semantic concepts can then be grouped together to form the DA that represents the sentence. Usually an effective approach, the rules are domain-specific and require multiple iterations of modification before achieving adequate coverage (Young, 2002). More advanced grammars such as Combinatorial Categorical Grammars (CCGs) (Steedman, 2000) were also applied to SLU. Learning to deal with ungrammatical spontaneous speech and erroneous recognition output requires relaxed grammars (Nguyen et al., 2006; Zettlemoyer and Collins, 2007). The major advantage of using these parsing-based methods is that they can readily model long-range dependencies, which is important when applied to complex language where concepts may be split up in the utterance, e.g., by relative clauses.

Modern SLU methods can be roughly grouped into two classes, one with internal sequential labels at the word-level or one with only a sentence level label. The first class of methods usually requires an alignment between the words in the input utterance and the target semantics, while the second one does not. Most of the sequential labelling SLU adopt the BIO tag scheme, which provides a method of aligning spans of words with labels. For example, an input utterance "*Seven Days is a nice Chinese restaurant in the north .*" can be labelled as "*B-restaurant I-restaurant O O O B-foodtype O B-area I-area I-area .*", where "*B*" indicates the word is at the beginning of the concept, "*I*" inside the concept, and "*O*" outside the concept, respectively. On the other hand, if the label is at the sentence level, it can just simply be the DA representation of the sentence such as "*inform(name=Seven Days, food=chinese, area=north)*". Based on the input utterance, and its corresponding labels, we can then apply machine learning algorithms to learn the SLU components directly from data.

To frame SLU as a machine learning problem, we can either use a generative model or a discriminative model. Given the input x and label y, generative models learn a joint probability distribution $\mathrm{P}(\mathrm{x},\mathrm{y})$ and then assign labels based on the derived conditional distribution $\mathrm{P}(\mathrm{y}|\mathrm{x})$. Dynamic Bayesian Network (DBN) were widely applied to SLU by modelling the semantics of the input utterance as a joint probability distribution of the hidden variable with the observed words (He and Young, 2006; Levin, 1995; Miller et al., 1994). Moreover, to conquer the challenge of modelling long-term dependencies between words imposed by the Markovian assumption, some approaches proposed to mitigate dependencies by introducing a hierarchical hidden structure (He and Young, 2006; Miller et al., 1996) in the generation process.

Discriminative models, on the other hand, directly learn the conditional probability distribution $\mathrm{P}(\mathrm{y}|\mathrm{x})$ of labels given a feature representation of the input utterance. Unlike generative models that rely on independence assumptions over the feature set, discriminative models are not constrained by these assumptions and therefore any useful feature can be directly included in the models. Empirical studies have shown discriminative models tend to outperform generative models on the SLU task (Wang and Acero, 2006). Among many popular discriminative approaches, such as Markov Logic Network (Meza-Ruiz et al., 2008), Support Vector Machines (Kate and Mooney, 2006; Mairesse et al., 2009), Conditional Random Fields (CRF) (Lafferty et al., 2001) are a particularly popular choice for SLU (Wang and Acero, 2006; Zhou and He, 2011). CRFs have also been used to create associations with the word confusion network (Tur and Deoras, 2013), to jointly classify utterance topics (Jeong and Geunbae Lee, 2008) or combine with Convolutional Neural Network (CNN)s for joint intent detection and slot filling (Celikyilmaz and Hakkani-Tur, 2015).

Recently, progress has been made by applying discriminative neural network approaches to language comprehension. Since the input sentence has variable length, Recurrent Neural Networks (RNNs) and Convolutional Neural Networks (CNNs) are the two most common choices for sentence encoding. Methods that adopt RNN for sentence encoding typically read words one by one from the beginning of the sentence; depending on the label type, it can either predict a label for each time step, like sequential labelling (Mesnil et al., 2015; Yao et al., 2014), or predict a sentence-level label based on the encoded sentence vector. Although RNNs can potentially capture global sentence meanings, CNNs (Kalchbrenner et al., 2014; Kim, 2014) are the most reasonable sentence encoding method because they can capture both global and local sentence semantics by applying several levels of convolution and pooling operations. Convolution filters allow the model to make predictions based on the locally-extracted features while ignoring the global context whenever it is necessary. Mrkšić et al. (2016b) has employed CNNs as the feature extractor of a set of binary semantic classifiers and explored pre-trained word embedding's to improve the semantic decoding of dialogue systems. Recently, Rojas Barahona et al. (2016) also showed that the feature representations of a CNN can be weighted by the ASR confidence score of the input hypothesis to produce a more robust semantic decoder. Furthermore, CNN-based semantic decoders have also been studied for zero-shot intent expansion (Chen et al., 2016).

2.3 Dialogue State Tracking

In a single turn conversation scenario, such as when answering questions, the output of the semantic decoder should provide enough information to fully encode the user's request. However, if multi-turn conversations are expected, the *dialogue state* should be tracked, from turn to turn, to accumulate the necessary data needed to make system decisions. Therefore, the term *dialogue state* loosely denotes a full-representation of the information the system needs to take its next action. For example, in a slot filling dialogue system, the dialogue state is typically defined by a list of user-provided constraints. When dealing with erroneous ASR output, a distribution of possible dialogue states is usually maintained instead of a point estimate. This distribution is sometimes referred to as the *belief state* of the dialogue (Rapaport, 1986). The five Dialogue State Tracking Challenges (Henderson et al., 2014a,b; Kim et al., 2016a,b; Williams et al., 2013) (DSTC 1$\sim$5) have sparked interest in dialogue state tracking. Baseline methods like the *focus* tracker use a set of handcrafted rules to update the dialogue state given observations provided by the SLU (Smith, 2014; Wang and Lemon, 2013).

Generative and discriminative models can be applied to track dialogue states (like SLU does). Generative DST approaches generally follow the *Hidden Information State* model (Young et al., 2010) and use a Dynamic Bayesian Network to model the joint distribution of the observed input semantics (the output of SLU) and the hidden dialogue state. Although many variants exist (Lee et al., 2014; Williams, 2010), one of the most representative approaches is the Bayesian Update of Dialogue State (BUDS) system (Thomson and Young, 2010).

Instead of learning the joint distribution, discriminative approaches for DST directly model the conditional probability distribution of dialogue states, given the input semantic labels, from the SLU output. The parameters are then updated by maximising the conditional log-likelihood. In the first few DSTCs, the most popular discriminative methods were the Maximum Entropy and neural network models (Henderson et al., 2013; Lee and Eskenazi, 2013; Ren et al., 2014; Sun et al., 2014). However, neural network approaches (Henderson et al., 2014c; Mrkšić et al., 2016b; Perez, 2016) gradually dominated the field because of their flexibility and strong discriminative power. Furthermore, recent research in DST also shows a tendency to integrate DST and SLU, and model the pair with a single neural network (Henderson et al., 2014c; Mrkšić et al., 2015). Although these end-to-end DST methods tend to produce models with better performance and mitigate the error propagation problem, they sacrifice the interpretability of the system and impede debugging.

2.4 Dialogue Management

Based on the dialogue state inferred from the DST component, the system's dialogue manager must decide what to say next. Conventional dialogue systems typically maintain one hypothesis of the dialogue state and make system decisions based on this point estimate (e.g., flowchart-based systems (Lucas, 2000; Sutton et al., 1996), form filling systems (Goddeau et al., 1996), and systems that makes decisions based on logical inference and planning (Larsson and Traum, 2000)). However, none of these approaches suggest a systematic way of learning or delineate the actions needed for each dialogue state. Thus, the dialogue manager is one of the many handcrafted components of the system.

By casting the decision-making step as a Markov Decision Process (MDP), we can learn the action selection model directly from interactions (Levin and Pieraccini, 1997; Singh et al., 1999). However, learning action selection based on a point estimate of the dialogue state is not ideal because ASR errors create unreliable results. Therefore, the Partially Observable Markov Decision Process (POMDP) formulation of dialogue systems (Young, 2002) offer a more robust and well-founded framework for statistical dialogue modelling.

In a POMDP-based statistical dialogue system framework, the ASR, SLU, and DST components provide multiple output hypotheses to encode recognition uncertainty. A probability distribution, rather than a point estimate, of the dialogue state is then used as the input of the dialogue management component. Therefore, the dialogue manager can be cast as a POMDP, which considers the distribution over many dialogue states when selecting the next system action (Roy et al., 2000b; Williams and Young, 2005). If we define the dialogue state as s and the action as a, then the belief state is a probability distribution over the possible states $\mathrm{b} = \mathrm{P(s)}$. The mapping π of a state to an action is denoted as the policy of the system. Learning the policy can be facilitated by defining a reward function $\mathrm{r(s,a)}$ for each state-action pair and by optimising the parameters against the expected accumulated reward (or total return)

$$R = \sum_t r(s_t, a_t) \tag{2.1}$$

This learning paradigm is called Reinforcement Learning (RL) (Sutton and Barto, 1998) because good actions are reinforced with positive rewards during training.

There are two major approaches for applying RL to dialogue management, the value- and policy-based methods. In value-based RL, a value function $\mathrm{Q(s,a)}$ is the expected future reward for taking action a in state s. Once the Q function is learned, the optimal policy can be extracted by applying the maximum operator

$$\pi(s) = \operatorname*{argmax}_a Q(s,a) \tag{2.2}$$

Recently, Gaussian Process (GP) has been used to model Q functions (Gasic and Young, 2014) for policy learning in dialogue systems. This technique has shown impressive efficiency in policy optimisation, which can be used to learn from real human subjects (Gašić et al., 2013) via direct interaction. On the other hand, policy-based methods (or policy gradients) parameterise the policy as a conditional probability distribution of action given state $\pi = \mathrm{P(a|s)}$ and optimise π directly against the reward function. Though optimisation is typically harder, and less efficient (Jurčíček et al., 2011), policy-based methods are attractive because they are natural for combining different learning paradigms (Silver et al., 2016; Su et al., 2016a).

2.5 Natural Language Generation

Natural Language Generation (NLG) is the final module in the dialogue system pipeline. Its main responsibility is to transform the semantic representation offered by the dialogue manager into natural language. Since language is the primary medium for communicating between the user and dialogue system, NLG is critical because it has a significant impact on usability and perceived quality. However, using data-driven NLG for SDS applications remains relatively unexplored due to the lack of an effective algorithm that can learn from structured input-output pairs and the difficulty of collecting semantically aligned corpora (Dušek and Jurcicek, 2015). Most existing NLG approaches commonly employ rules and heuristics (Cheyer and Guzzoni, 2007; Mirkovic and Cavedon, 2011); they also tend to generate rigid and stylised responses without the natural variation of human language. Moreover, these limitations significantly add to development cost and slow down cross-domain, cross-lingual dialogue system deliveries, by making them complex and expensive.

In a pipelined dialogue system setting, the NLG is simplified to a module whose task is to transform the information encoded in the semantic representation output by the dialogue manager (Gasic and Young, 2014; Young et al., 2013) into a human-readable text. Therefore, the system does not need to handle content selection but only sentence planning and surface realisation (Konstas and Lapata, 2012; Walker et al., 2002). Typically, the NLG takes a DA representation as its input and outputs one or multiple sentences in natural language. Since there are many ways to define a set of DAs, this thesis mainly works with the DAs described in Appendix B. For example a DA such as "*inform(name=seven_days, food=chinese)*" could be transformed to either "*Seven days serves Chinese food*" or "*There is a Chinese restaurant called Seven days*". Since there is a one-to-many mapping between the input and outputs, building a good NLG system is nontrivial because this typically means finding a sweet spot that can strike a balance between language fluency and variation. Due to this tradeoff, supervised learning for NLG is not always an apparent choice, while reinforcement learning has been shown to have a potential to tackle this problem (Rieser and Lemon, 2011).

The following of this section reviews three existing NLG methods across different application domains. There is one rule-based approach and two statistical methods that incorporate machine learning to different degrees. The HALOGEN system (Langkilde and Knight, 1998) can be viewed as the first statistical NLG approach where a language model was employed to re-rank the output of a rule-based generator. The SParKY framework (Stent et al., 2004; Walker et al., 2002) suggests a systematic approach by dividing the problem into pipelines and applying machine learning techniques in different submodules. This background establishes the foundation for the introduction of a family of more data-driven NLG approaches we called the corpus-based methods in Chapter 3.

2.5.1 Template-based Language Generator

By definition, template-based NLG systems are natural language generating systems that map non-linguistic semantic input to the linguistic surface structure without intermediate representations (Reiter and Dale, 2000; Van Deemter et al., 2005). Adopting an example from Reiter and Dale (2000), a simple template-based system might associate its semantic input form

$$\mathrm{Departure}(\mathrm{train} = 306, \mathrm{location} = Aberdeen, \mathrm{time} = 10:00) \tag{2.3}$$

directly with a template such as

$$\text{[train] is leaving [town] now,} \tag{2.4}$$

and then fill in the corresponding values of each individual slot via a database search

$$\text{The train is leaving Aberdeen now.} \tag{2.5}$$

Note that this template will only be used when the time referred to is close to the intended time of speaking. Other templates must be used to generate departure expressions that relate to the past or future. In practice, a hierarchical top-down approach is adopted in template-based NLG systems; the final surface form is generated recursively and composed of several individual sub-rules. Example rules of the template-based generator used in our spoken dialogue systems are shown in Appendix C.

Although building a template-based NLG system is relatively straightforward, maintaining and updating its rules becomes more and more challenging (Reiter and Dale, 1997) as the target application becomes more complex. Furthermore, most template-based NLG systems do not incorporate linguistic insights (Busemann and Horacek, 1998) or statistical variations; they tend to produce rather rigid and stylised responses without the natural variation found in human language.

2.5.2 Linguistically Motivated Approaches

Unlike template-based approaches, which directly associate a template with an input semantic representation, most of the mainstream, linguistically-motivated NLG methods rely on an indirect mapping; the input semantic is mapped to an intermediate representation and then back to the surface form. For example, this type of NLG system will not directly map Equation 2.3 to Equation 2.4; instead, Equation 2.3 will map to

$$\text{Leave}_{present}(\text{train} = \text{train}_{demonstrative}, \text{location} = Aberdeen, \text{time} = now), \tag{2.6}$$

where lexical items and style of reference have been determined while linguistic morphology is still absent. Although the details may vary, linguistically-motivated NLG systems can start from the same semantic representation. Then, the system undergoes several consecutive transformations where various NLG submodules can operate until a final surface form is produced.

The SPaRKy Generation Framework

The SPaRKy (Sentence Planning with Rhetorical Knowledge) framework (Stent et al., 2004) is an extension of SPoT (Sentence Planning Trainable) (Walker et al., 2002), which is extended via the addition of Rhetorical Structure Theory (RST) (Mann and Thompson, 1988). Based on this framework, the NLG problem can be divided into three modules connected in a pipeline: sentence plan generation, sentence plan re-ranking, and surface realisation. SPaRKy, the sentence planner gets the content plan selected by the dialogue manager (also referred to as the content planner) and applies a sentence plan generator to produce one or multiple sentence plans. Each sentence plan, which is represented by a set of text plan trees (tp-trees) (Stent et al., 2004), consists of a set of DAs need to communicate the rhetorical relations that hold between them. Then, a sentence plan re-ranker takes the sentence plans and selects the most promising one as the input of the surface realiser. The surface realiser converts the chosen sentence plan into a natural language surface form by replacing each of the leaf nodes with a word from the lexicon.

Sentence Plan Generation The basis of the sentence plan generator is a set of clause-combining operations that operate on tp-trees and incrementally transform elementary representations called DSyntS (Melčuk, 1988) that are associated with the DAs on the leaves of the tree. The output of the sentence plan generation component contains two structures: (1) a tp-tree, which is a binary tree with leaves labelled by the assertions from the input tp-tree and interior nodes labelled with clause-combing operations; and (2) one or more DSyntS trees, which reflect the parallel operations on the predicate-argument representations. During the generation phase, the sentence plan generator samples several valid sentence plans based on the input tp-tree. Most of the sentence plan generators in the literature are purely handcrafted and therefore their scalability is an issue. Although Stent and Molina (2009) proposed learning sentence planning rules directly from a corpus of utterances labelled with Rhetorical Structure Theory (RST) discourse relations (Mann and Thompson, 1988), the required corpus labelling is expensive; additional handcrafting is still needed to map the sentence plan to a valid syntactic form.

Sentence Plan Re-ranking In the original SPaRKy paper (Stent et al., 2004), the sentence plan re-ranker is the core component that employs machine learning. Based on the set of tp-trees produced by the generator, the sentence plan re-ranker learns its ranking rules via a labelled set of sentence plan training examples using the RankBoost algorithm (Schapire, 1999). Each input tp-tree is pre-processed and subsequently represented by a feature vector. This feature vector is extracted by handcrafting a variety of templates and rules based on the tree structure. On the other hand, each input tp-tree is fed into the RealPro surface realiser (Lavoie and Rambow, 1997) to generate a realisation. This realisation is then presented to an operator, who assigns a label with a number from 1 to 5 as an indicator of the realisation's quality. The objective is then to rank the loss,

$$\mathrm{F(x)} = \sum_{\mathrm{i}} \alpha_{\mathrm{i}} \mathrm{f_i(x)} \tag{2.7}$$

$$\mathcal{L}(\mathrm{x,y}) = \sum_{\mathrm{x,y} \in \mathrm{D}} \mathrm{exp}^{-(\mathrm{F(x)}-\mathrm{F(y)})}, \tag{2.8}$$

where x, and y are tp-trees, x is preferred over y, $\mathrm{f_i}(\cdot)$ are the functions that generate the feature vector, α_{i} are the model parameters, and D is the entire dataset.

Surface Realisation The surface realiser then takes the selected sentence plan and converts it into natural language. Typical surface realisers involve at least three kinds of processing methods: the first is syntactic realisation, where grammar rules are used to choose inflections, add function words, or decide the order of the components; the second is morphological realisation, which computes inflected forms; and finally, the third is orthographic realisation, which deals with casing, punctuation, and formatting. These are very basic steps, but most realisers are capable of considerably more complex processing (Espinosa et al., 2008; Lavoie and Rambow, 1997; White et al., 2007). Recently, research on learning statistical surface realisation rules from data have been proposed (Cuayahuitl et al., 2014; Dethlefs et al., 2013).

The HALOGEN system

The HALOGEN system, developed by Langkilde and Knight (1998), is another example of a linguistically-motivated NLG approach. Like the SPaRKy framework, HALOGEN also employs over-generation and re-ranking strategies. In the over-generation phase, a symbolic generator produces several candidate realisations in a word-lattice form-based on a set of grammar rules. These candidate sentences are then re-ranked by an n-gram LM, trained via a collection of news articles. This method can be considered as the first step in combining a symbolic generator with a statistical re-ranker. It has been successfully applied to a specific dialogue system domain (Chambers et al., 2004). However, like all the other linguistically motivated generators, the major drawback of this method is that the performance still heavily relies on a generator. Generators are rule-intensive; machine learning is only applied on the part of the system (such as re-ranking) that is relatively indirect to the actual performance.

2.6 Conclusion

In summary, although dialogue system research covers a wide range of topics and has been studied extensively across several communities, the learnability of NLG is relatively underexplored. Previous popular approaches are either template-based or grammar-based methods that employ a bunch of rules to transform the original meaning representations given by the dialogue manager to the corresponding linguistic surface forms. Despite the fact that grammar-based approaches are a good fit to model complex linguistic phenomenon such as aggregations and discourse relations, little of these sophisticated responses are required in a real-world dialogue system deployment. In contrast, what we need is a rather lean approach that can model just enough characteristics of human language but readily scalable to other domains and languages if provided with sufficient training data. This thesis focuses on studying NLG approaches that integrate statistical and machine learning methods to improve output quality and system scalability.

Chapter 3

Corpus-based Natural Language Generation for Dialogue Systems

Although template and grammar-based approaches have been dominating the NLG research community in the past decade, *corpus-based* methods (Mairesse et al., 2010; Mairesse and Young, 2014; Oh and Rudnicky, 2000) have received more and more attention as access to data becomes increasingly available. Here we refer to this line of NLG approaches as *corpus-based* methods as it usually involves learning to generate or select a response from a corpus of sentences. By defining a flexible learning structure, *corpus-based* methods aim to learn generation decisions directly from data. Compared to grammar-based systems such as SPaRKy and HALOGEN, which assume hierarchical tree-like generation processes, most of the corpus-based approaches embrace a much flatter model design such as a language-model-based sequential generation model. Learning directly from data enables the system to mimic human responses more naturally, removes the dependency on predefined rules, and makes the system easier to build and extend to other domains. However, these approaches suffer from the inherent computational cost gained in the over-generation and re-ranking phases (Oh and Rudnicky, 2000). This computational cost increases further when semantic errors in the rendered output cannot be tolerated. The approaches also have difficulty collecting sufficient training data with fine-grained semantic alignments (Dušek and Jurcicek, 2015).

3.1 Class-based Language Model

Proposed by Oh and Rudnicky (2000), the class-based language model is the first pure statistical generation model used in dialogue systems. Unlike previous methods, class-based models have an n-gram language model for a generator. It produces sentences word by word and is based on the language model probability

$$p(w_t|w_{j-1},...w_0,c) \approx p(w_t|w_{j-1},...w_{j-n},c), \tag{3.1}$$

where w$_*$ is a word, c is the class, and n is the number of steps predicted by the Markov assumption. Therefore, the probability of the sentence W can be written down as a joint probability distribution of all the words that compose the sentence

$$p(\mathbf{W}|c) = \prod_j p(w_j|w_{j-1},...w_{j-n},c). \tag{3.2}$$

Consequently, if we sample words one-by-one based on Equation 3.1 until an end-of-sentence token is generated or a predefined length is reached, we can produce a sentence from the model. To control the meaning of the generated sentences, researchers partitioned the training corpus into several small clusters based on the dialogue act (DA) of each individual sentence. The partition criteria is the DA type and a few slots[1]. The sentences in the same class are then used to train an n-gram LM for that class. During the generation phase, based on the input DA, the corresponding class LM is selected to over-generate a set of candidate outputs, which a rule-based re-ranker can rescore from. Various penalties are assigned to the candidate sentences in the rescoring phase if the sentence

- is either to short or too long (determined by class dependent thresholds),
- contains repetitions of any of the slots,
- contains slots whose values are null in the DA,
- does not have required slots.

Although class-based LM generation provides a way for learning language generation directly from data, there are several drawbacks to this approach: firstly, the sentence class partitions are crude, which impedes model generalisation (e.g., unseen combination of DA types, slot-value pairs); secondly, the Markovian assumption of the n-gram LM prevents the model from learning long-range dependencies; finally, last but not least, the generation is extremely inefficient (Mairesse and Young, 2014; Wen et al., 2015a) (e.g., thousands of candidate sentences need to be produced in the over-generation phase in order to achieve a reasonable performance). Although the generation process is straightforward and easy to implement, it is difficult to apply in practice due to its defects.

---

[1] In the original paper, Oh and Rudnicky (2000) partitioned the corpus based on only one slot. However, in follow-up work by (Mairesse and Young, 2014; Wen et al., 2015a), the authors both show that partitioning based on more slots yields better results.

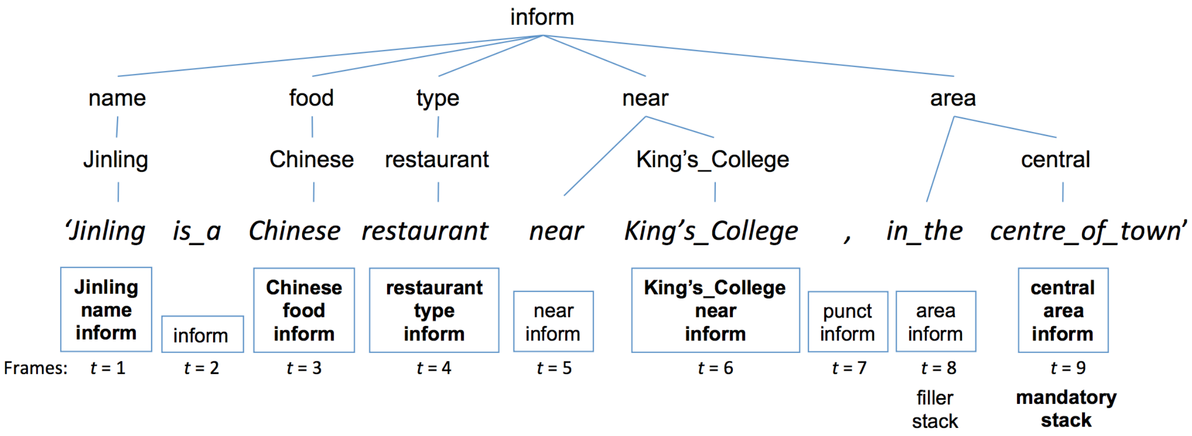


Fig. 3.1 An example of the sentence for the "*inform*" DA and its corresponding stack representation and tree equivalent used in Mairesse and Young (2014). Figure borrowed from Mairesse and Young (2014).

3.2 Phrase-based Factored Language Model

Motivated by both linguistic theories and statistical sequential generation models, BAGEL, which is proposed by Mairesse and Young (2014) is a phrase-based generator built on the Dynamic Bayesian Network framework. As shown in Figure 3.1, BAGEL uses a set of stacks to approximate and represent the tree-structure-like nature of human language. Each stack is either mandatory or functional. A mandatory stack creates phrase-based content using semantic information provided by DA input. A functional stack creates functional phrases that transform semantics into sentences. By modelling semantics and phrases as a sequence of stacks, NLG is cast as a two-sequence decoding problem, one on the semantic level and the other on the phrase level.

Based on the DA input, represented by a set of unordered mandatory stacks $\mathrm{S_m}, |\mathrm{S_m}| \leq \mathrm{J}$, the goal is to find a sequence of phrases $\mathbf{R} = (\mathrm{r_1}, \mathrm{r_2}, ...\mathrm{r_J})$ such that the probability of the realisation given the DA $\mathrm{P}(\mathbf{R}|\mathrm{S_m})$ is maximised,

$$\mathbf{R}^* = \underset{\mathbf{R}}{\operatorname{argmax}} P(\mathbf{R}|S_m). \tag{3.3}$$

We can then rewrite the objective function by introducing two additional variables: (1) $\mathbf{S}_\mathrm{m}$, an ordered mandatory stack sequence, and (2) $\mathbf{S}$, a full stack sequence created from the

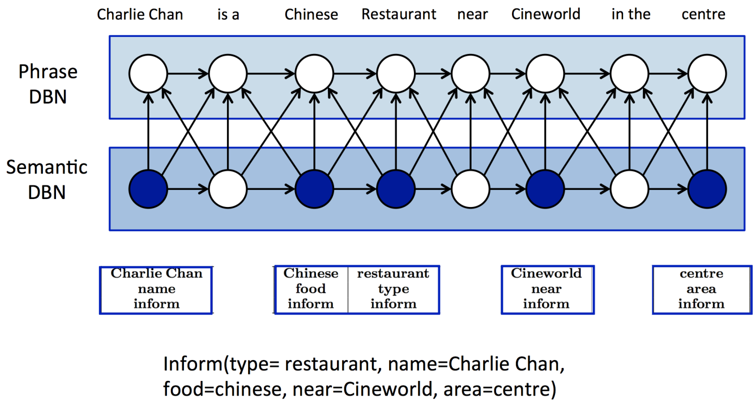


Fig. 3.2 An overview of the phrase-based DBN model for NLG. The lower level is a semantic DBN consisting of both mandatory (blue) and functional stacks (white), while the upper level is a phrase-based DBN that oversees mapping semantic stacks to phrases.

functional stack set.

$$\begin{aligned}
\mathrm{P}(\mathbf{R}|S_m) &= \sum_{\mathbf{S}_m}\sum_{\mathbf{S}} \mathrm{P}(\mathbf{R},\mathbf{S},\mathbf{S}_m|S_m) && (3.4)\\
&= \sum_{\mathbf{S}_m}\sum_{\mathbf{S}} \mathrm{P}(\mathbf{R}|\mathbf{S},\mathbf{S}_m,S_m)\mathrm{P}(\mathbf{S}|\mathbf{S}_m,S_m)\mathrm{P}(\mathbf{S}_m|S_m) && (3.5)\\
&= \sum_{\mathbf{S}_m} \mathrm{P}(\mathbf{S}_m|S_m)\sum_{\mathbf{S}} \mathrm{P}(\mathbf{R}|\mathbf{S})\mathrm{P}(\mathbf{S}|\mathbf{S}_m), && (3.6)
\end{aligned}$$

where we assume conditional independence given $\mathbf{S}_m, S_m \perp\!\!\!\perp \mathbf{R} \mid \mathbf{S}$ and $S_m \perp\!\!\!\perp \mathbf{S}|\mathbf{S}_m$. However, inference of such a model would require a summation over all possible underlying stack sequences and all possible realisations; therefore, this model is not practical in real-time applications such as dialogue systems. Therefore, the authors adopted a greedy approach by breaking down the inference into the following three steps:

1. A content ordering model that chooses the optimal stack order $\mathbf{S}_m^* = \mathrm{argmax}_{\mathbf{S}_m} \mathrm{P}(\mathbf{S}_m|S_m)$,
2. A content planning model that inserts functional stacks in between mandatory stacks and outputs the full stack sequence $\mathbf{S}^* = \mathrm{argmax}_{\mathbf{S}} \mathrm{P}(\mathbf{S}|\mathbf{S}_m)$.
3. A surface realising model that converts each stack into a phrase $\mathbf{R}^* = \mathrm{argmax}_{\mathbf{R}} \mathrm{P}(\mathbf{R}|\mathbf{S})$

The first two models can be viewed as a DBN for modelling semantics, while the last one is another DBN for modelling phrases. The process is illustrated in Figure 3.2.

Although phrase-based DBN is promising, and the authors showed that it can achieve better performance than the class-based n-gram LM approach, the main limitation of the method is the requirement of additional semantic alignment labels between the semantic components and phrases. Mairesse et al. (2010) have proposed to mitigate this additional labelling load by introducing active learning into the framework. The labelling is nevertheless difficult to collect since the annotations are not intuitive; thus, the labellers still need to be trained beforehand.

3.3 Example-based Approaches

Another popular corpus-based method for NLG is the example-based approaches, in which the generation process is cast as a template extraction and ranking problem. For example, Angeli et al. (2010) trained a set of log-linear models to develop a series of generation decisions that chose the most suitable template from a corpus of sentence examples. Kondadadi et al. (2013) later showed that the output could be further improved by an SVM re-ranker, making the output comparable to human-authored text.

The k-Nearest Neighbour (*knn*) approach is an example-based method for NLGs. The core idea is very similar to the template extraction method proposed by Angeli et al. (2010). During training, the DA is pre-processed and represented by a one-hot vector, as described in Section 5.1. The sentence is delexicalised and mapped into a template-like structure. In the testing phase, the testing DA $\mathbf{d}_\mathrm{i}$ is processed similarly and used to compute cosine similarity against each of the examples $\mathbf{d}_\mathrm{j}$ in the training set

$$sim(\mathbf{d}_i, \mathbf{d}_j) = \frac{\mathbf{d}_i \cdot \mathbf{d}_j}{|\mathbf{d}_i||\mathbf{d}_j|}. \tag{3.7}$$

This similarity score is then used to select the best matching template T^* for realising the sentence

$$T^* = F(\operatorname*{argmax}_{\mathbf{d}_j} sim(\mathbf{d}_i, \mathbf{d}_j)), \tag{3.8}$$

where $\mathrm{F}(\cdot)$ is a one-to-one mapping from the DA to the template. In cases where there are multiple templates matching to the same DA, we randomly sample one as the target. Then, a post-processing lexicalisation is performed by inserting appropriate slot values in input DA into the template placeholders.

3.4 Attention-based Sequence-to-Sequence Approaches

(<root> <root> ((X-name n:subj) be v:fin ((Italian adj:attr) restaurant n:obj (river n:near+X))))
X-name n:subj be v:fin Italian adj:attr restaurant n:obj river n:near+X

Fig. 3.3 Examples adopted from Dušek and Jurcicek (2016). Trees encoded as sequences used for the seq2seq generator (top) and the reranker (bottom).

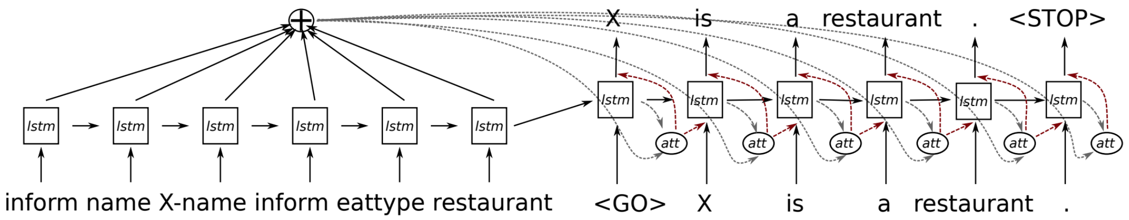


Fig. 3.4 Figure adopted from Dušek and Jurcicek (2016). The proposed attention-based seq2seq generator architecture.

Recently, deep learning-based approaches have became popular in the NLG community. Two of most the similar works to part of this thesis are Mei et al. (2015) and Dušek and Jurcicek (2016), which are both based on a sequence-to-sequence (or encoder-decoder) architecture combined with an attention mechanism. Since Dušek and Jurcicek (2016)'s work is more relevant to this thesis (also a dialogue NLG problem), this section focuses on introducing their work.

The idea of the work is to check whether a deep syntax tree (Dušek and Jurcicek, 2016) encoding of the output sentences can help to improve the generation performance. The proposed model architecture employs the idea of over-generation and reranking idea and includes a generator as shown in Figure 3.4 and a reranker in Figure 3.5. The job of the generator is to take the input DA encoded in a sequence of semantic components and generate a sequence of tokens that represent a sentence in its linearised deep syntax tree representation, as shown in Figure 3.3. After a bunch of sentences are over-generated from the generator, a reranker (Figure 3.5) is used to rescore the generated sentences by mapping them back to DA and adding penalty if the resulting DA is not equal to the original one.

Although encoding the output sentence using a linguistically-motivated structure like deep syntax tree is a good idea, their result showed that this additional linguistic structure only helps when a reranker is not used. This suggests that although the deep syntax tree can inject additional heuristics into the model, the model can just be better off by including an additional machine learned reranker. Moreover, the BAGEL dataset (Mairesse and Young, 2014) used in the experiment was pretty small (around 2k sentences) and the improvement of the added heuristic rules maybe simply come from data sparsity.

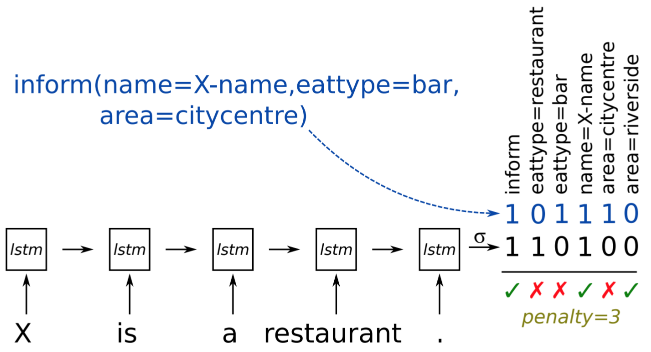


Fig. 3.5 Figure adopted from Dušek and Jurcicek (2016). The proposed reranker architecture.

In Section 5.3, an attention-based encoder-decoder is introduced and compared to other methods proposed in this thesis. The implementation of that model basically adopted the lesson learned from both Mei et al. (2015) and Dušek and Jurcicek (2016).

3.5 Evaluation Metrics and Difficulties

Evaluating NLG systems is almost as hard as building them. The major difficulty comes from the fact that there is no single metric that can properly tell whether a generated sentence is a good one. As noted in Stent et al. (2005), a good generator usually depends on the following four factors:

1. Adequacy: to be adequate, a sentence needs to express the *exact meaning* of the input *unambiguously*.
2. Fluency: a sentence is only fluent if it is *grammatically correct* and *idiomatic*.
3. Readability: a sentence is readable if it is both adequate and fluent *in a context*.
4. Variability: a generator needs to be able to produce multiple sentences that fulfil the adequacy, fluency and readability constraints.

The fourth factor is the most difficult of the non-statistical approaches because it literally means handcrafting more templates or rules for each input DA.

3.5.1 BLEU score

Corpus BLEU One of the popular metrics used in NLG evaluations is the BLEU score (Papineni et al., 2002). The BLEU score first became popular in the Machine Translation (MT) community, where it is used to compute the similarity between a candidate and a set of reference sentences,

$$\text{BLEU} = \text{BP} \cdot \exp(\sum_{n=1}^{N} w_n \log p_n), \tag{3.9}$$

where n is the n-gram length up to N, $\mathrm{w_n}$ are positive weights summing to 1, and $\mathrm{p_n}$ are the geometric average of the modified n-gram precisions

$$p_n = \frac{\sum_{C \in \{\text{Candidates}\}} \sum_{\text{ngram} \in C} \text{Count}_{\text{clip}}(\text{ngram})}{\sum_{C' \in \{\text{Candidates}\}} \sum_{\text{ngram}' \in C} \text{Count}(\text{ngram}')}, \tag{3.10}$$

where C is the candidate sentence, $\text{Count}(\cdot)$ is the normal counting function and $\text{Count}_{\text{clip}}(\cdot) = \max(\text{Count}(\cdot), \text{Max\_Ref\_Count})$ is the truncated counting function. The set of equations suggest that we can truncate each n-gram's count, if necessary, to not exceed the largest count observed in any single reference for that n-gram. This term basically serves as a rough measure of the n-gram similarity between the candidate and the reference set. Note that we sum over the candidate set {Candidates}, therefore BLEU is a corpus-level metric rather than at the sentence-level. Because n-gram models prefer short sentences, we need an additional term called brevity penalty (BP) as in Equation 3.9 to penalise these overly-short sentences

$$\text{BP} = \begin{cases} 1 & \mathrm{c} > \mathrm{r} \\ \exp^{(1-\mathrm{r}/\mathrm{c})} & \mathrm{c} \leq \mathrm{r} \end{cases} \tag{3.11}$$

where r is the length of the best match reference (in terms of length) and c is the length of the candidate. Therefore, this penalty only exists if the generated sentence is shorter than any of the other references.

In Papineni et al. (2002), the authors showed that the BLEU score correlates with human perceived quality on MT tasks very well. However, studies applying it to NLG (Belz, 2006; Novikova et al., 2017; Stent et al., 2005) were not very positive; in fact, studies showed that the BLEU score, and other automatic metrics such as ROUGE (Lin and Hovy, 2003) or NIST (Doddington, 2002), either negatively correlate with human perceptions or only correlate slightly. Thankfully, this can be mitigated by having a large collection of references for each candidate (Belz, 2006). These automatic metrics can only serve as a metric when developing systems and real performance can only be properly accessed by human studies.

Sentence BLEU BLEU is usually computed at the corpus-level to assess the quality of a generation system. When computing it at the sentence-level, higher order n-grams are usually harder to match and result in very poor score. To mitigate this and come up a good approximate of the BLEU score at the sentence-level, He and Deng (2012) proposed a smoothed version of BLEU score – the sentence BLEU – where it modifies the original n-gram precisions $\mathrm{p_n}$ as,

$$p_n = \frac{\sum_{C\in\{\text{Candidates}\}}\sum_{\text{ngram}\in C}\text{Count}_{\text{clip}}(\text{ngram}) + \eta \cdot p_n^0}{\sum_{C'\in\{\text{Candidates}\}}\sum_{\text{ngram}'\in C}\text{Count}(\text{ngram}') + \eta}, \tag{3.12}$$

where η is a smoothing factor usually set to 5 and $\mathrm{p_n^0}$ is set by $\mathrm{p_n^0} = \mathrm{p_{n-1}} \cdot \mathrm{p_{n-1}}/\mathrm{p_{n-2}}$ for $\mathrm{n} \geq 3$. $\mathrm{p_1}$ and $\mathrm{p_2}$ are estimated empirically. The smoothing term in Equation 3.12 allows us to fallback the n-gram matching to lower order n-grams when higher order ones are not matched. Moreover, the brevity penalty is not clipped,

$$\text{BP} = \exp^{(1-r/c)} \tag{3.13}$$

Sentence BLEU in both previous literature (Auli and Gao, 2014; He and Deng, 2012) and this thesis are used only as an alternative objective to train models rather than a metric for the evaluation. During evaluation, the corpus BLEU is used.

3.5.2 Slot Error Rate

Adequacy can sometimes be captured very well if the application domain is simple. A spoken dialogue system where most of the responses are short and compact is an example. In this case, the slot error rate (Wen et al., 2015a,c), an evaluation metric based on exact string matching between the candidate surface form and a handcrafted semantic dictionary, can be used to calculate a coarse estimate of the generator's adequacy score. This metric was also used in Oh and Rudnicky (2000) for reranking sentences. The equation slot error rate ERR, over the entire corpus, is characterised by

$$\text{ERR} = \frac{\sum_i |s_{i,\text{missing}}| + \sum_i |s_{i,\text{redundant}}|}{\sum_i |s_i|}, \tag{3.14}$$

where $|\mathrm{s_i}|$ is the number of slots of the i-th example and $|\mathrm{s_{i,missing}}|$ and $|\mathrm{s_{i,redundant}}|$ are the counts of the missing and redundant slots, respectively. Note that the counting of $|\mathrm{s_{i,missing}}|$ and $|\mathrm{s_{i,redundant}}|$ is dependent on the exact string matching between a semantic dictionary and the output sentence. Note, slot error rate can only be treated as an approximate metric for assessing a generation system's adequacy when the domain is simple. However, when the domain becomes bigger and the generated sentences become more complex the use of slot error rate for assessing adequacy may become obscure. This thesis focuses on assessing NLG in domain-specific dialogue system scenarios and therefore slot error rate can still give a good approximation of the generation quality.

3.5.3 Language Variability

Evaluating language variability of a generation system is the most difficult part in NLG evaluation. Although there are quite a lot of works have been proposed in the past (Cao and Clark, 2017; Li et al., 2016a) to address this problem, none of the existing metrics correlates well enough to human perceptions. Instead of evaluating with existing metrics, this thesis proposes to assess the quality of an NLG system by evaluating its top-5 generated sentences. If all the top-5 sentences are scored high on the other metrics (BLEU and Slot Error Rate), the system can be considered as a good system that can generate unique and high quality responses.

3.6 Conclusions

This chapter has presented the corpus-based methods which aim at learning generation decisions directly from data. There approaches of this method have been introduced: the class-based n-gram language generator, the phrase-based DBN model, and the example-based kNN approach. Different approaches may require different level of annotations. Complex annotation schemes may limit the scalability of the model.

Since we are focusing on building dialogue systems that can easily scale to bigger domains once the data is provided, it is more attractive to focus on corpus-based NLG approaches because they employ few rules and minimal handcrafted components. Although this may mean that we could lose a part of the generalisation capabilities brought by the linguistic insights, training on large corpora can potentially grant the model a generalisation capability that is less arbitrary. In the long run, this could be a more general way to go.

In the following chapters, we are going to focus on corpus-based NLG methods and apply Recurrent Neural Network (RNN) LM (Mikolov et al., 2010) as a generator to replace the class-based n-gram model (Oh and Rudnicky, 2000). Unlike Mairesse and Young (2014), we will focus on models that can learn to generate from a paired DA-sentence corpus without the need of additional semantic alignment annotations.

Chapter 4

Neural Networks and Deep Learning

4.1 Neural Networks

Artificial Neural Networks (ANNs) were inspired and developed as mathematical models that mimic brain function (McCulloch and Pitts, 1988; Rumelhart et al., 1988). Although it is now clear that ANN processes bear little resemblance to neurological behaviour, they remain popular pattern classifiers. ANN is composed of many simple processing units called neurons joined together by weighted connections. The computational neuron is activated via a signal input, the activation spreads throughout the entire network along the weighted connections. The electrical activity of biological neurons typically follows a series of sharp *spikes*, while the activation of an ANN node was originally intended to model the average firing rate of these spikes.

One very basic type of ANN is the Feed-forward Neural Network (FNN), as shown in Figure 4.1. Each neuron is represented by a circle and each synapse by an arrow connecting two neurons. Notice that the direction of the connection is one-way, from input to output. The input pattern **x** is introduced to the input layer, then propagated through the hidden layer **h** and finally to the output layer **y**

$$\mathbf{h} = \sigma(\mathbf{W}_x\mathbf{x}), \tag{4.1}$$

$$\mathbf{y} = \phi(\mathbf{W}_h\mathbf{h}), \tag{4.2}$$

where $\mathbf{W}_{\mathrm{x}}$ and $\mathbf{W}_{\mathrm{o}}$ are weight matrices, $\sigma(.)$ and $\phi(.)$ are hidden and output layer activation functions. To achieve a greater expressive power and mimic the *spike* shape activation in the biological brain, $\sigma(.)$ is usually selected as a sigmoid function

$$g(x) = \frac{1}{1+e^{-x}}, \tag{4.3}$$

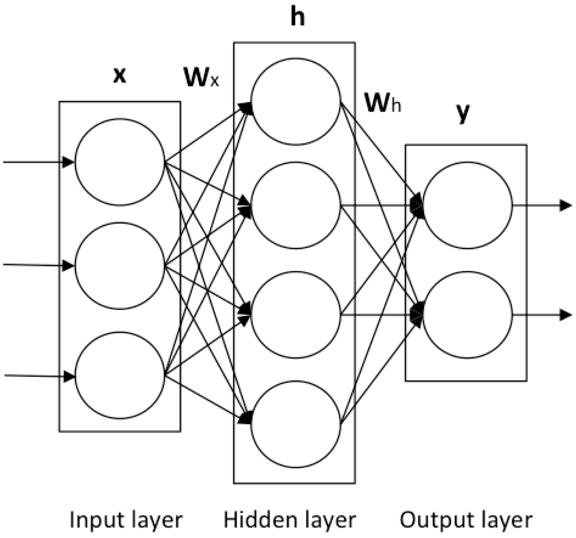


Fig. 4.1 A feed-forward Artificial Neural Network

or a hyperbolic tangent

$$\tanh(x) = \frac{e^{2x} - 1}{e^{2x} + 1} \tag{4.4}$$

The type of output layer activation depends on the task at hand. The most common approach to solve classification problems such as language modelling or sentiment classification is the soft-max function

$$\phi(x_i) = \frac{e^{x_i}}{\sum_j e^{x_j}}, \tag{4.5}$$

where i is the index of the output neuron. The soft-max activation effectively transforms the network output into a probability distribution over the labels. Up to now, the process of computing output, given an input, is known as the forward pass of the network.

Training an NN requires us to re-estimate the parameters (weight matrices) of the network to minimise the network loss. Although many possibilities exist for defining the loss function of a classification network, the most widely adopted today is the cross entropy error between the true target value **t** and the prediction **y**

$$\mathcal{L}(\theta) = -\mathbf{t}^{\mathsf{T}} \log(\mathbf{y}). \tag{4.6}$$

Because of the complex dependency relations, and the nonlinearity inside the network, finding an exact optimal solution is intractable. As a result, NNs are usually optimised using the gradient desent with error backpropagation method (Rumelhart et al., 1986; Werbos, 1988). The basic idea of gradient desent is to find the derivative of the loss function with respect to each of the network weights θ, then adjust the weights in the direction of the negative slope

$$\theta^{(t+1)} = \theta^{(t)} - \alpha^{(t)} \frac{\partial \mathcal{L}(\theta)}{\partial \theta^{(t)}}, \tag{4.7}$$

where $\alpha^{(\mathrm{t})}$ is the learning rate that typically decreases with training iteration t. If the loss function is computed with respect to the entire training set, the weight update procedure is referred to as batch learning. This contrasts with online learning, where weight updates are performed with respect to individual training examples only. In many studies (Lecun et al., 1998b), online learning tends to be more efficient than batch; especially when learning a large dataset where significant redundancy is used. In addition, the stochastic nature of online learning can help to escape from local minima (Lecun et al., 1998b). As a result, Stochastic Gradient Decent (SGD) is used in all the experiments in this thesis.

Backpropagation, on the other hand, is simply a repeated application of the chain rule for partial derivatives. The first step is to calculate the derivatives of the loss function with respect to the output units. Given the cross entropy loss $\mathcal{L}(\theta)$, its derivative with respect to an output unit before soft-max $\mathrm{a_o}$ is

$$\frac{\partial \mathcal{L}}{\partial a_o} = \sum_k t_k \frac{\partial \log y_k}{a_o} = -\sum_k t_k \frac{1}{y_k} \frac{\partial y_k}{\partial a_o}. \tag{4.8}$$

Since $\mathbf{y}$ is the output of a soft-max application on $\mathbf{a}$, we know

$$\frac{\partial y_k}{\partial a_o} = \begin{cases} y_o(1-y_o), & o = k. \\ -y_o y_k, & o \neq k. \end{cases} \tag{4.9}$$

After substituting Equation 4.8 by Equation 4.9 and reorganising, we arrive at

$$\frac{\partial \mathcal{L}}{\partial a_o} = y_o - t_o. \tag{4.10}$$

Now we continue to apply the chain rule, working backwards to compute the error derivative of the hidden layers. At this point, it will be helpful to introduce the following notation

$$\delta_j := \frac{\partial \mathcal{L}}{\partial a_j}, \tag{4.11}$$

where j is any unit in the network. For the units in the hidden layer, we have

$$\delta_h = \frac{\partial \mathcal{L}}{\partial v_h} \frac{\partial v_h}{\partial a_h} = \frac{\partial v_h}{\partial a_h} \sum_{o=1}^{O} \frac{\partial \mathcal{L}}{\partial a_o} \frac{\partial a_o}{\partial v_h} = \sigma'(a_h) \sum_{o=1}^{O} \delta_o \theta_{ho}, \tag{4.12}$$

where O is the size of output layer, $\mathrm{a_h}$ and $\mathrm{v_h}$ are the value of hidden units h before and after passing through the activation function σ. Once the δ terms for all hidden units are obtained, we can calculate the error derivative with respect to each network's weight

$$\frac{\partial \mathcal{L}}{\partial \theta_{ij}} = \frac{\partial \mathcal{L}}{\partial a_j}\frac{\partial a_j}{\partial \theta_{ij}} = \delta_j v_i. \tag{4.13}$$

These gradients are then used to update weights as described in Equation 4.7. The procedure of propagating gradients backwards from the output nodes to the input nodes is known as the backward pass of the network.

4.1.1 Recurrent Neural Networks

In the previous section, we considered feed-forward neural networks that have no cycles in the connection. If we allow feedback cycles inside the network, we obtain Recurrent Neural Networks (RNNs). Many variants of RNN have been proposed, such as the Elman network (Elman, 1990), Jordan network (Jordan, 1990), and the echo state network (Jaeger, 2001). The RNN we refer to in this report is the Elman RNN containing a single, self-connected hidden layer, as shown in Figure 4.2. The major difference between RNN and FNN is that FNN can only map an input vector to an output vector, whereas an RNN can, in principle, map from the entire history of the inputs into each output. This property makes RNN the best model structure for sequences. Theoretically, an RNN, with enough hidden units, is an universal approximator that can learn to map any two arbitrary-length sequences to a certain accuracy (Hammer, 1998). The key point is that the recurrent connection allows the hidden units to "memorise" previous inputs and thereby influence the output of the network.

The forward pass formula for RNNs is like FNN; the only difference is the hidden layer activation. To consider previous hidden units in each time step, Equation 4.1 is changed to

$$\mathbf{h}_t = \sigma(\mathbf{W}_x\mathbf{x}_t + \mathbf{W}_r\mathbf{h}_{t-1}), \tag{4.14}$$

where t denotes the time step. $\mathbf{W}_\mathrm{r}$ is the newly introduced weight matrix representing the recurrent connection.

To perform the backward pass, two well-known algorithms have been proposed to efficiently calculate the derivative of RNN parameters: real time recurrent learning (Robinson and Fallside, 1987) and Back-propagation through time (BPTT) (Werbos, 1990). The BPTT algorithm is adopted in this report because it is conceptually simpler and computationally more efficient. Given an unrolled view of RNN, shown in Figure 4.3, the BPTT algorithm,

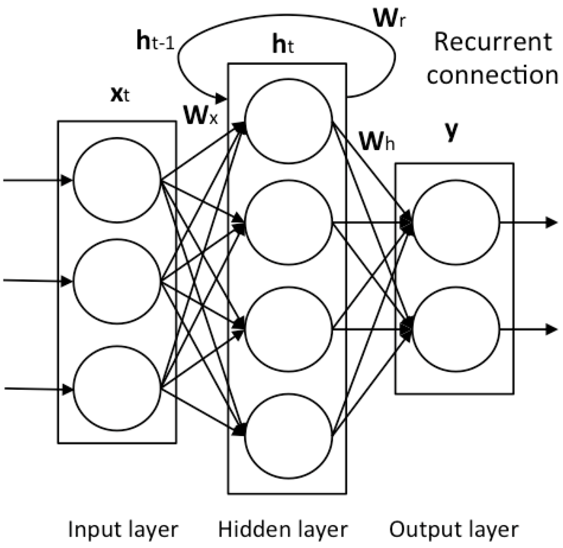


Fig. 4.2 Architecture of the Elman recurrent neural network. The recurrence is fully connected between two hidden layers in adjacent time steps.

like standard backpropagation, also consists of repeated application of the chain rule. The difference is that the loss function depends on the activation of the hidden layer in RNN; thus, the influence occurs on both the hidden and output layers for each single time step. Therefore,

$$\delta_h^t = \sigma'(a_h^t)\left(\sum_{o=1}^{O} \delta_o^t \theta_{ho} + \sum_{h'=1}^{H} \delta_{h'}^{t+1} \theta_{hh'}\right), \tag{4.15}$$

where

$$\delta_j^t := \frac{\partial \mathcal{L}}{\partial a_j^t}. \tag{4.16}$$

The sequence of δ terms can then be calculated recursively using Equation 4.15; beginning with $\delta_{\mathrm{j}}^{\mathrm{T}+1} = 0$ for all j from the end of the sequence. Finally, since the weights are shared for each time step, we need to sum over the entire sequence to get derivatives with respect to the network parameters,

$$\frac{\partial \mathcal{L}}{\partial \theta_{ij}} = \sum_{t=1}^{T} \frac{\partial \mathcal{L}}{\partial a_j^t} \frac{\partial a_j^t}{\partial \theta_{ij}} = \sum_{t=1}^{T} \delta_j^t v_i^t \tag{4.17}$$

4.1.2 Long Short-term Memory

As noted in the previous session, an important benefit of RNN is the ability to use arbitrarily long context information when mapping between input and output sequences. In practice, the range of context that a vanilla RNN can process is quite limited. The reason is that the influence of a given input on the hidden layer, and therefore on the network output, either

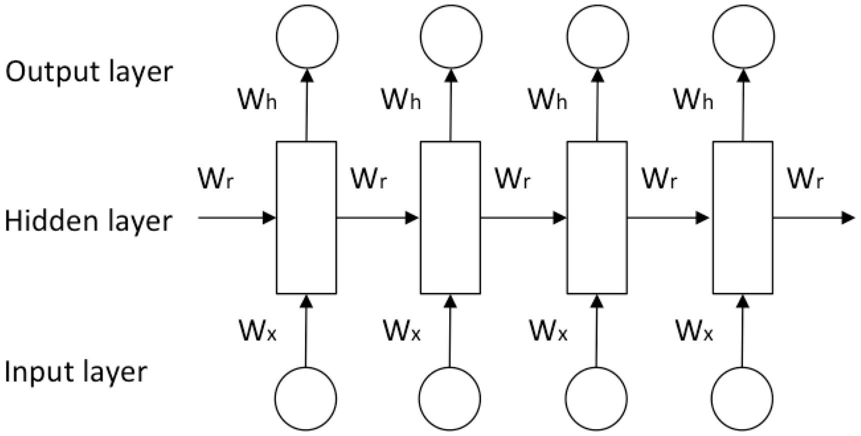


Fig. 4.3 An unfolded view of RNN. Each rectangle represents a layer of hidden units in a single time step. The weighted connections from input to hidden layer are $\mathbf{W}_\mathrm{x}$, those from hidden to output are $\mathbf{W}_\mathrm{h}$, and the hidden to hidden weights are $\mathbf{W}_\mathrm{r}$. Note that the same weights are reused at every time step.

decays or grows exponentially as it cycles around the network's recurrent connections. This phenomenon is usually referred to as the *vanishing gradient problem* in the literature (Bengio et al., 1994; Hochreiter et al., 2001). The vanishing gradient problem is illustrated schematically in Figure 4.4.

Many methods were proposed to overcome the vanishing gradient problem in RNN. Some methods include simulated annealing and discrete error propagation (Bengio et al., 1994), hierarchical sequence compression (Schmidhuber, 1992), and Gated Recurrent Units (Chung et al., 2014). We apply the Long Short-term Memory (LSTM) (Hochreiter and Schmidhuber, 1997) architecture in this thesis. We choose this popular method for sequence modelling because it has been tested on several similar problems, e.g., speech recognition (Graves et al., 2013a), handwriting recognition (Graves et al., 2009), spoken language understanding (Yao et al., 2014), and MT (Sutskever et al., 2014).

Although multiple memory blocks in a cell are allowed, Figure 4.5 provides an illustration of an LSTM cell with only one memory block. The memory block is recurrently connected and can be used to store historical information. Each memory block is associated with three multiplication units, the input, forget, and output gates. The multiplication units provide the cell with the continuous analogies of write, read, and reset. These multiplication gates allow LSTM memory cells to store and access information over long periods, thereby mitigating the vanishing gradient problem. For example, if the input gate remains closed (activation→0), the value of the memory block won't be overwritten by the new input in the network, and can therefore be made available to the net much later in the sequence, by opening the output gate.

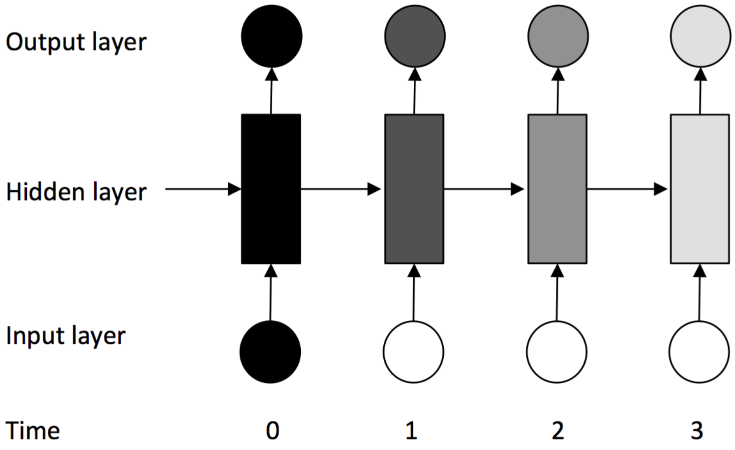


Fig. 4.4 In the RNN vanishing gradient problem, the colour of nodes is determined by the sensitivity of a single node to input at time 0. Due to the repeated multiplication of the recurrent weights, the influence of the input will vanish or grow over time. The network will eventually forget the first input.

To compute the forward pass of an LSTM, the gate activation level need to be calculated that given the notation defined in Figure 4.5

$$\mathbf{i}_{\mathrm{t}} = \sigma(\mathbf{W}_{\mathrm{xi}}\mathbf{x}_{\mathrm{t}} + \mathbf{W}_{\mathrm{hi}}\mathbf{h}_{\mathrm{t}-1}) \tag{4.18}$$
$$\mathbf{f}_{\mathrm{t}} = \sigma(\mathbf{W}_{\mathrm{xf}}\mathbf{x}_{\mathrm{t}} + \mathbf{W}_{\mathrm{hf}}\mathbf{h}_{\mathrm{t}-1}) \tag{4.19}$$
$$\mathbf{o}_{\mathrm{t}} = \sigma(\mathbf{W}_{\mathrm{xo}}\mathbf{x}_{\mathrm{t}} + \mathbf{W}_{\mathrm{ho}}\mathbf{h}_{\mathrm{t}-1}), \tag{4.20}$$

where σ is a sigmoid function. The proposed cell value at the current time step is therefore

$$\hat{\mathbf{c}}_t = tanh(\mathbf{W}_{xc}\mathbf{x}_t + \mathbf{W}_{hc}\mathbf{h}_{t-1}). \tag{4.21}$$

Combining this equation with the input and forget gate obtained earlier, we can calculate the new cell value via element-wise multiplication of the gate and cell values as in

$$\mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \hat{\mathbf{c}}_t. \tag{4.22}$$

Finally, the hidden layer representation is computed by multiplying the output gate and cell value element-wise as in

$$\mathbf{h}_t = \mathbf{o}_t \odot tanh(\mathbf{c}_t). \tag{4.23}$$

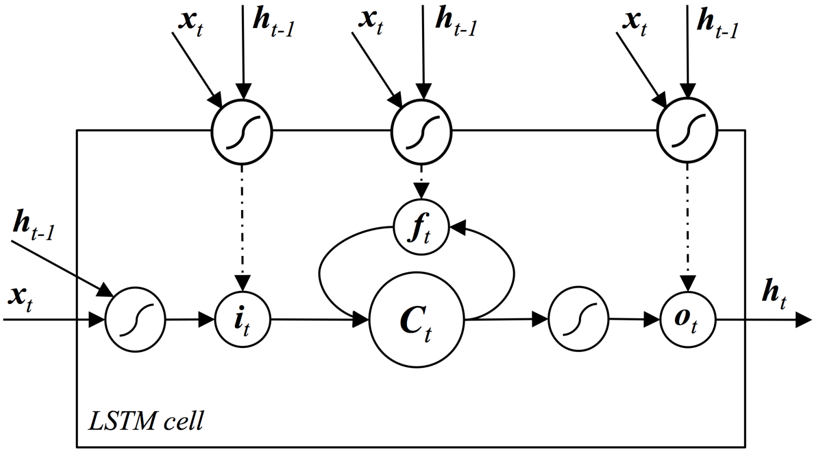


Fig. 4.5 A Long short-term memory cell is an RNN architecture with the memory block $\mathbf{C}_t$. $\mathbf{i}_t$, $\mathbf{f}_t$,$\mathbf{o}_t$ are input, forget, and output gates. $\mathbf{x}_t$ and $\mathbf{h}_{t-1}$ are controlling signals of the three gates. Typically $\mathbf{x}_t$ is the current time step's input and $\mathbf{h}_{t-1}$ is the hidden layer's representation at previous time step.

The LSTM cell is meant to replace the original additive hidden layer of vanilla RNNs. Consequently, once we obtain the hidden layer representation, we can substitute it into the original RNN output layer; all the other parts of the forward pass work the same.

Since an LSTM is a differentiable function approximator like an RNN, gradient-based methods applied to RNNs can also be used to train LSTMs. Therefore, BPTT is also adopted for calculating the gradients.

4.1.3 Bidirectional Networks

In this section, we consider both the forward and backward context of a sequence classification problem because of the benefits it may provide. For example, given two sentences:

1. Seven days is *a* expensive restaurant.
2. Seven days is *an* expensive restaurant.

To decide whether to choose *a* or *an*, we only need to consider the word after the indefinite article rather than the previous words. An obvious solution would be to add a time-window of future context to the network input. However, the range of context the input can consider is limited and is often decided by heuristics. This in turn constrains the expressive power of the network and may also impose an asymmetric bias of the two contexts.

Bidirectional RNNs (Baldi et al., 1999; Schuster and Paliwal, 1997a) offer a more elegant solution. Instead of using only the forward context, the network encodes the input sequence

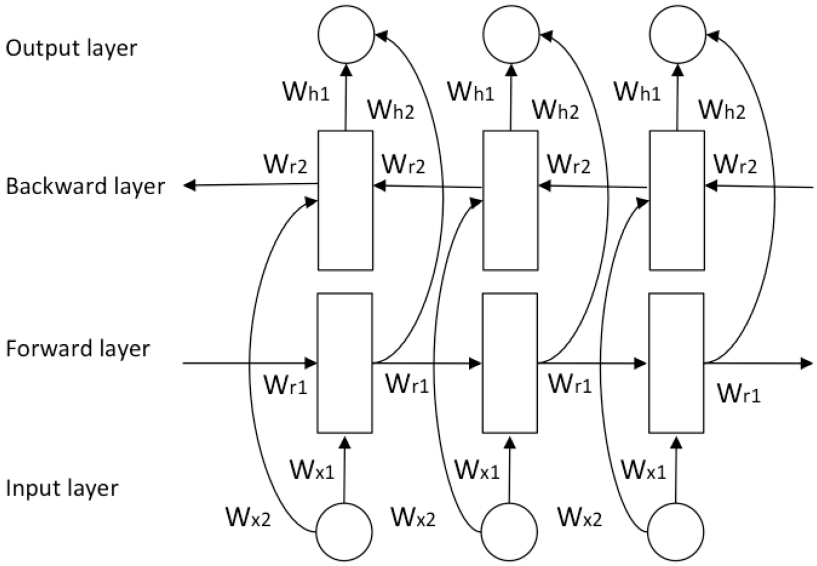


Fig. 4.6 The unfolded bidirectional RNN. The backward layer and forward layer can encode information from both directions. Six sets of distinct weights are included. Note there are no connections between two hidden layers.

in both directions by two separate RNNs and connects both the forward and backward hidden layers to a single output layer, as can be seen in Figure 4.6. This structure provides the output layer with context based on the future and the past at every point in the input sequence, without displacing the inputs from the relevant targets. Bidirectional networks (Schuster and Paliwal, 1997b) have been shown to be effective for sequential problems such as protein secondary structure prediction (Chen and Chaudhari, 2004), speech recognition (Graves et al., 2013b), and MT (Sundermeyer et al., 2014).

The forward pass of the network is the same as an RNN, the only difference is that the input sequence is also presented in the opposite direction so that the output layer can only be updated when both the forward and backward hidden layers are obtained. Similarly, the backward pass is initiated like the RNN trained with BPTT; the only difference is that the gradients of the output layer are computed first, then fed back to the two RNNs.

4.1.4 Convolutional Neural Networks

The Convolutional Neural Network (CNN) is a biologically-inspired variant of NNs. According to very early research on the visual cortex (Hubel and Wiesel, 1968), we know that the visual cortex is composed of a complex arrangements of cells that are sensitive to small sub-regions of the visual fields (receptive fields). The sub-regions are tiled to cover the entire

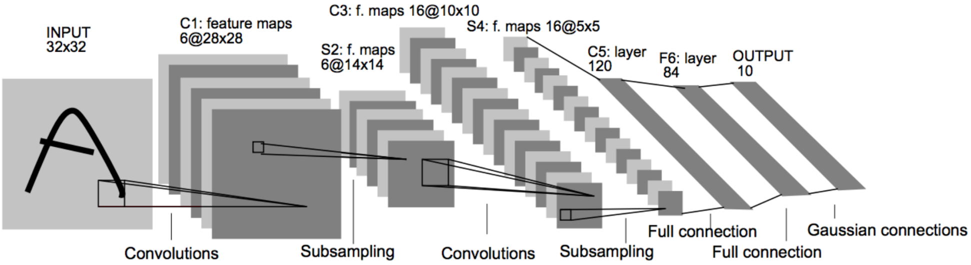


Fig. 4.7 Architecture of a CNN model called LeNet-5 (Lecun et al., 1998a). Each plane is a feature map to capture certain patterns of the previous layer. A fully connected FNN is attached on top as the classifier for recognising input digits.

visual field. These cells act as local filters over the input space and are well-suited to exploit the strongly spatial local correlations present in natural images.

CNNs combine three architectural concepts to ensure some degree of shift, scale, and distortion invariance: local receptive fields, shared weights, and spatial or temporal subsampling, as shown in Figure 4.7. The inputs of the LeNet CNN (Lecun et al., 1998a) are pixels of the target image that are classified and arranged as a matrix according to their location in the image. A set of local filters (2D weighted connections) is then applied to aggregate pixels in the filter window and forms a set of feature maps. The convolution operation is applied by sliding each of the local filters through the entire input image. Then, a weighted value of input pixels and weights for each location are determined. With these local receptive fields, neurons can extract elementary visual features such as oriented edges, end-points, and corners. The subsequent layers then combine these features to detect higher order features. Another benefit of the convolutional approach is that, by applying the same weights across the entire image, the feature detector can extract similar features in different parts of the image. Thus the feature detector is capable of handling slight distortions or translational features in the image. Typically, a CNN contains multiple feature detectors for each layer to allow multiple features to be captured at the same location. This convolutional process is the basis of the robustness of CNNs, which allows shifts and distortions of the input.

Once a feature has been detected, its exact location becomes less important. Only its approximate position relative to other features is relevant. The precise location may in fact be harmful because the positions are likely to vary for different instances of the same label. This is done by introducing a spatial or temporal subsampling layer between two convolutional layers, as shown in Figure 4.7. Two well-known subsampling techniques are average pooling and maximum pooling. Although maximum pooling is believed to be more discriminative and computationally efficient in most cases, average pooling seems to work better for certain problems (Wen et al., 2015a).

After several convolution-pooling operations, a set of features maps is extracted. A fully-connected FNN is then attached on top of the final feature layer and takes the extracted feature maps as an input to determine the label of the image. Another crucial advantage of the pooling operation is that it can transform a variable-size image into a fixed-size vector representation so that it can be fed into the FNN for classification. Since all the weights in the CNN can be learned by standard backpropagation, the CNN is thus synthesising its own feature extractor. This is important since we can remove the process of designing feature templates in the traditional pattern recognition pipeline, which is obviously time-consuming and not scalable. We let the network figure out the important features on its own.

CNNs were first studied in the computer vision literature on topics such as object recognition (Lecun et al., 1998a) and handwriting recognition (Ciresan et al., 2011). Recently, these methods have been also adopted in problems such as speech recognition (Sainath et al., 2013). The convolutional sentence model (Kalchbrenner et al., 2014; Kim, 2014) uses the same methodology but collapses the two-dimensional convolution and pooling process into a single dimension. The resulting model is claimed to represent the state-of-the-art for many NLP related tasks (Kalchbrenner et al., 2014; Kim, 2014).

4.2 Objective Function

Neural networks are mathematical models that, after trained, can be used to make predictions on unseen input. During training time, given a labelled dataset $\mathrm{D} = \{\mathrm{x_i}, \mathrm{y_i}\}$, the goal is to tune the set of available model parameters θ so that a paramterised function of input $\mathrm{f}_\theta(\mathrm{D})$ is maximised or minimised. This function $\mathrm{f}_\theta(\cdot)$ is called the objective function of the optimisation procedure. Two objective functions: (1) Maximum Likelihood Estimation, which is used everywhere in the thesis, and (2) Discriminative Training, which is used in the domain adaptation in Chapter 6, are introduced below.

4.2.1 Maximum Likelihood

Maximum Likelihood Estimation (MLE) is probably the most common objective function in modern machine learning. As the name implies, the objective function of MLE is usually defined as the likelihood function of the dataset $\mathrm{p}(\mathrm{D})$. This likelihood function in supervised setting is usually interpreted as a conditional probability of label $\mathrm{y_i}$ given input $\mathrm{x_i}$,

$$\theta_{MLE} = \underset{\theta}{\operatorname{argmax}} \prod_i p(y_i|x_i) \tag{4.24}$$

As taking a product of some numbers less than one would approach zero as the number of those numbers goes to infinity, we can instead work in the log space. Therefore, instead of optimising Equation 4.24, an alternative objective function is,

$$\theta_{MLE} = \underset{\theta}{\operatorname{argmax}} \log \prod_i p(y_i|x_i) = \underset{\theta}{\operatorname{argmax}} \sum_i \log p(y_i|x_i) \tag{4.25}$$

4.2.2 Discriminative Training

In contrast to the MLE criteria, whose goal is to maximise the log-likelihood of correct examples, Discriminative Training (DT) objective aims at separating correct examples from competing incorrect examples. Given a training instance $(\mathrm{x_i}, \mathrm{y_i})$, the training process starts by generating a set of model predictions $\mathrm{GEN}(\mathrm{x_i})$ of label $\mathrm{y_i}$ using the current model parameter θ and input $\mathrm{x_i}$. The discriminative objective function can therefore be written as,

$$\theta_{DT} = \underset{\theta}{\operatorname{argmax}} \sum_{\hat{y}_i \in GEN(x_i)} p_\theta(\hat{y}_i|x_i) R(\hat{y}_i, y_i) \tag{4.26}$$

where $\mathrm{R}(\hat{\mathrm{y}}_\mathrm{i}, \mathrm{y_i})$ is the scoring function evaluating candidate $\hat{\mathrm{y}}_\mathrm{i}$ by taking ground truth $\mathrm{y_i}$ as reference. $\mathrm{p}_\theta(\hat{\mathrm{y}}_\mathrm{i}|\mathrm{x_i})$ is the scaled distribution of the candidate and is typically computed as

$$\mathrm{p}_\theta(\hat{\mathrm{y}}_\mathrm{i}|\mathrm{x_i}) = \exp\left(\gamma \log \mathrm{p}(\hat{\mathrm{y}}_\mathrm{i}|\mathrm{x_i}, \theta)\right) / \textstyle\sum_{\hat{\mathrm{y}}_\mathrm{j} \in \mathrm{GEN}(\mathrm{x_i})} \exp\left(\gamma \log \mathrm{p}(\hat{\mathrm{y}}_\mathrm{j}|\mathrm{x_i}, \theta)\right). \tag{4.27}$$

$\gamma \in [0, \infty]$ is a tuned scaling factor that flattens the distribution for $\gamma < 1$ and sharpens it for $\gamma > 1$. Note, the un-normalised candidate likelihood $\log \mathrm{p}(\hat{\mathrm{y}}_\mathrm{t}|\mathrm{x_i}, \theta)$ is the model prediction.

Since the DT objective presented here is differentiable everywhere w.r.t. model parameters θ, optimisation algorithms like back-propagation can be applied to calculate the gradients and update the parameters directly. The DT criterion introduced here can make use of the data at hand better than the MLE criterion. This is mainly because the objective function allows the model to explore its own limitations by refining its predictions via both positive and negative examples. This enables the model to correct its own mistakes, while adjusting towards the correct examples.

4.3 Stochastic Neural Networks

Deterministic Neural Networks, such as MLPs and RNNs, are popular models that apply well to regression and classification tasks. For classification tasks, deterministic NNs model the conditional distribution $\mathrm{p(y|x)}$ of the predictor variable y given the input variable x. This conditional distribution is deterministic because its shape solely depends on the input variable x and there is a single best model prediction $\mathrm{y^* = argmax_y\, p(y|x)}$ for each input. This is not the case in many real-world problems such as image or dialogue response generation. For example, given a user's input query, there are many responses that are valid and appropriate. Therefore, a strong deterministic model would fail to capture these multiple modes in dialogue responses and only learns to produce the most frequent and generic responses (Cao and Clark, 2017). One way to tackle this multi-modality in large learning systems is to introduce a latent variable and make it stochastic. The resulting framework is thus a Stochastic Neural Network. This is because the output is no longer only deterministic; thus it no longer only depends on the input, but also the randomness injected by the latent variables as well.

Due to recent advances in Neural Variational Inference (NVI) (Mnih and Gregor, 2014), the Variational Autoencoder (VAE) framework (Kingma and Welling, 2014; Rezende et al., 2014) has become more and more popular. The Variational Autoencoder (VAE) framework is a practical training technique for NN-based generative models with latent variables. We are more interested in its extension – Conditional VAE (cVAE) (Doersch, 2016) – because of it allows the output to be additionally conditioned on the input rather than drawn independently from a prior distribution.

4.3.1 Conditional Variational Autoencoder Framework

Based on an input x and output y, our goal is to model the conditional distribution of the output y, given the input x, with the introduction of a latent variable z for

$$p(y|x) = \int_z p_\theta(y|z) p_\theta(z|x) dz, \tag{4.28}$$

where $\mathrm{p_\theta(y|z)}$ and $\mathrm{p_\theta(z|x)}$ are both modelled by neural networks. Similar to VAE, cVAE modifies Equation 4.28 by replacing the conditional distribution $\mathrm{p_\theta(z|x)}$ with an approximate posterior distribution $\mathrm{q_\phi(z|x,y)}$. By parameterising the latent distribution, based on both x and y, $\mathrm{q_\phi(z|x,y)}$ is a powerful surrogate of $\mathrm{p_\theta(z|x)}$ during inference.

If cVAE were trained with a standard cross entropy objective, the model would learn to encode its inputs deterministically by making the variances of $\mathrm{q_\phi(z|x,y)}$ decay (Raiko et al., 2014). Instead, during VAE inference, we introduce the variational lower bound of the true-likelihood of the data as an alternative objective function to encourage the model to keep its posterior distribution close to the original conditional distribution $p_\theta(z|x)$,

$$\begin{aligned}\mathcal{L}(\theta,\phi) &= \mathbb{E}_{q_\phi(z|x,y)}[\log p_\theta(y|z)] - D_{KL}(q_\phi(z|x,y)||p_\theta(z|x))\\ &\leq \log \int_z p_\theta(y|z)p_\theta(z|x)dz\\ &= \log p_\theta(y|x) \qquad (4.29)\end{aligned}$$

Consequentially, from the $q_\phi(z|x,y)$ point of view, $p_\theta(z|x)$ serves as a regularisation term to encourage a multimodal distribution of the learnt posterior term. While in the $p_\theta(z|x)$ viewpoint, $q_\phi(z|x,y)$ is in the teacher network, from which it must learn.

4.3.2 Neural Variational Inference

To have a tighter variational lower bound, as shown in Equation 4.29, a common practise in the deep learning community is to parameterise and approximate the intractable, true posterior using a neural network $q_\phi(z|x,y)$, with a parameter set ϕ. Therefore, Neural Variational Inference (NVI) (Miao et al., 2016) begins by constructing an expressive inference network $q_\phi(z|x,y)$ based on the following steps

1. Construct vector representations of the observed variables $\mathbf{x} = f_x(x)$, $\mathbf{y} = f_y(y)$. Depending on the nature of the variables, the embedding functions $f_x(\cdot)$ and $f_y(\cdot)$ can be any kind of neural network that is suitable for data such as CNNs or RNNs.

2. Assemble a joint representation for $\mathbf{o} = g(\mathbf{x},\mathbf{y})$. $g(\cdot)$ is typically modelled via an MLP, by taking the concatenation of the two vector representations as input.

3. Parameterise the variational distribution over the latent variable $q_\phi(z|x,y) = l(\mathbf{o})$. $l(\cdot)$ is another neural network that can project the vector representation into a distribution.

The projection function $l(\cdot)$ can take on different forms ,depending on the type of latent variables we want to model. Most of the VAE literature employs a continuous latent variable and parameterise the latent distribution as a diagonal Gaussian, represented by its parameterised mean and standard deviation, as in $\mu = l_1(\mathbf{o}), \log\sigma = l_2(\mathbf{o})$. $l_1(\cdot)$ and $l_2(\cdot)$ are linear transformations, which output the parameters of the Gaussian distribution. However, if a discrete latent variable is considered for the latent distribution, then $l(\cdot)$ could be an MLP that maps the vector representation into a multinomial distribution or a set of Bernoulli distributions. As a result, by sampling from the variational distribution $z \sim q_\phi(z|x,y)$, we can carry out stochastic back-propagation and optimise the variational lower bound.

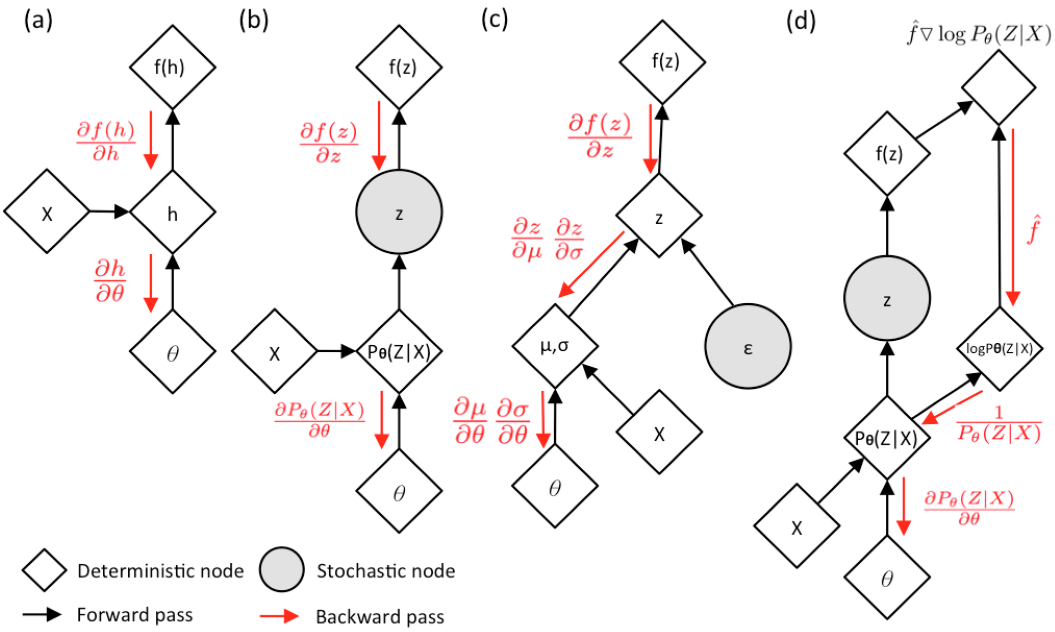

(a)
f(h)
$\frac{\partial f(h)}{\partial h}$
X
h
$\frac{\partial h}{\partial \theta}$
θ
(b)
f(z)
$\frac{\partial f(z)}{\partial z}$
z
X
Pθ(Z|X)
$\frac{\partial P_\theta(Z|X)}{\partial \theta}$
θ
(c)
f(z)
$\frac{\partial f(z)}{\partial z}$
z
$\frac{\partial z}{\partial \mu}$ $\frac{\partial z}{\partial \sigma}$
μ,σ
ε
$\frac{\partial \mu}{\partial \theta}$ $\frac{\partial \sigma}{\partial \theta}$
X
θ
(d)
$\hat{f}\nabla \log P_\theta(Z|X)$
f(z)
$\hat{f}$
z
logPθ(Z|X)
Pθ(Z|X)
$\frac{1}{P_\theta(Z|X)}$
$\frac{\partial P_\theta(Z|X)}{\partial \theta}$
X
θ
Deterministic node
Stochastic node
Forward pass
Backward pass

Fig. 4.8 Gradient estimation in stochastic computation graphs. Figure adapted from "Categorical Re-parameterization with Gumbel-Softmax" by Jang et al. (2016). (a) the gradient $\partial f(h)/\partial h$ can be directly computed via back-propagation if h is deterministic and differentiable. (b) The introduction of a stochastic node z precludes back-propagation because the sampling function $z^{(n)} \sim p_\theta(z|x)$ doesn't have a well-defined gradient. (c) The re-parameterisation trick (Kingma and Welling, 2014; Rezende et al., 2014), which creates an alternative gradient path to circumvent the sampling function inside the network, allows the gradient to flow from $f(z)$ to $\boldsymbol{\theta}$. Here we used the Gaussian re-parameterisation trick as an example, but other types of distributions can also be applied. (d) The scoring function-based gradient-estimator obtains an unbiased estimate of $\nabla f(z)$ by back-propagating along a surrogate loss $\hat{f}\nabla \log p_\theta(z|x)$, where $\hat{f} = f(z) - b(z)$ and $b(z)$ is the baseline for variance reduction.

As shown in Figure 4.8(b), the introduction of the stochastic latent variable z precludes the back-propagation period because the sampling function $z \sim p_\theta(z|x)$ doesn't have a well-defined gradient[1]. Therefore, to carry out variational inference in stochastic neural networks, one can either use the re-parameterisation trick as shown in Figure 4.8(c) or the scoring function-based method in Figure 4.8(d).

Instead of sampling directly inside the network and blocking the gradient flow as in Figure4.8(b), the re-parameterisation trick (Figure 4.8(c)) (Kingma and Welling, 2014; Rezende et al., 2014) moves the internal stochastic node out of the computation graph and

---

[1] In practise, we sample from the variational distribution $z \sim q_\phi(z|x,y)$ which doesn't have a well-defined gradient either.

treats it as a random input node ($\mathbf{z}$ in Figure 4.8(b) $\rightarrow$ ε in Figure 4.8(c)). Therefore, the encoder network does not produce a latent distribution $\mathrm{p}_\theta(\mathrm{z}|\mathrm{x})$ directly. Instead, the network only outputs the parameters for constructing the target distribution, for example, the mean μ and variance σ^2 of a Gaussian distribution. We can then reconstruct the sample $\mathbf{z}^{(\mathrm{n})}$ via the expression

$$\begin{aligned} \mathbf{z}^{(\mathrm{n})} &= \mu + \sigma \cdot \varepsilon \\ \varepsilon^{(\mathrm{n})} &\sim \mathrm{N}(0,1). \end{aligned} \tag{4.30}$$

In this way, the gradient can be directly back-propagated through the entire network. This is because the stochastic node has been moved out of the graph. The re-parameterisation trick is commonly combined with a Gaussian random variable to model a continuous latent distribution inside a neural network. Recently, a Gumbel-Softmax re-parameterisation trick (Jang et al., 2016) was also proposed to approximate the categorical distribution of discrete latent-variable models.

The scoring function-based estimator (also referred to as REINFORCE (Williams, 1992) or the likelihood ratio estimator (Glynn, 1990)) is illustrated in Figure 4.8(d). It computes the gradient based on the following derivation

$$\begin{aligned} \nabla_\theta \mathbb{E}_{\mathrm{p}_\theta(\mathrm{z}|\mathrm{x})}[\mathrm{f}(\mathrm{z})] &= \int \nabla_\theta \mathrm{p}_\theta(\mathrm{z}|\mathrm{x})\mathrm{f}(\mathrm{z})\mathrm{dz} && (4.31) \\ &= \int \frac{\mathrm{p}_\theta(\mathrm{z}|\mathrm{x})}{\mathrm{p}_\theta(\mathrm{z}|\mathrm{x})} \nabla_\theta \mathrm{p}_\theta(\mathrm{z}|\mathrm{x})\mathrm{f}(\mathrm{z})\mathrm{dz} && (4.32) \\ &= \int \mathrm{p}_\theta(\mathrm{z}|\mathrm{x}) \nabla_\theta \log \mathrm{p}_\theta(\mathrm{z}|\mathrm{x})\mathrm{f}(\mathrm{z})\mathrm{dz} && (4.33) \\ &= \mathbb{E}_{\mathrm{p}_\theta(\mathrm{z}|\mathrm{x})}[\mathrm{f}(\mathrm{z}) \nabla_\theta \log \mathrm{p}_\theta(\mathrm{z}|\mathrm{x})] && (4.34) \end{aligned}$$

where from Equation 4.32 to Equation 4.33 we apply the log derivative trick,

$$\nabla_\theta \log p_\theta(z|x) = \frac{\nabla_\theta p_\theta(z|x)}{p_\theta(z|x)}. \tag{4.35}$$

Therefore, based on Equation 4.34, we can approximate this expectation using a Monte Carlo method by first drawing samples from the conditional distribution $\mathrm{z}^{(\mathrm{n})} \sim \mathrm{p}_\theta(\mathrm{z}|\mathrm{x})$ and then computing the weighted gradient term

$$\mathbb{E}_{p_\theta(z|x)}[f(z)\nabla_\theta \log p_\theta(z|x)] \approx \frac{1}{N}\sum_{n=1}^{N} f(z^{(n)})\nabla_\theta \log p_\theta(z^{(n)}|x). \tag{4.36}$$

This unbiased estimator of the gradient does not require $\mathrm{f(z)}$ to be differentiable. Instead, we just need to be able to evaluate or observe its value for a given z. However, the real challenge is that this Monte Carlo gradient estimator typically has a high variance. To make it useful, we must ensure that its variance is as low as possible. Additionally, to gain more control over the variance, we can subtract a control variate $\mathrm{b(z)}$ from the learning signal $\mathrm{f(z)}$. We add back its analytical expectation $\mu_\mathrm{b} = \mathbb{E}_{\mathrm{p}_\theta(\mathrm{z}|\mathrm{x})}[\mathrm{b(z)}\nabla_\theta \log \mathrm{p}_\theta(\mathrm{z}|\mathrm{x})]$ to keep it unbiased

$$\nabla_\theta \mathbb{E}_{p_\theta(z|x)}[f(z)] = \mathbb{E}_{p_\theta(z|x)}[(f(z) - b(z))\nabla_\theta \log p_\theta(z|x)] + \mu_b \tag{4.37}$$

4.4 Conclusion

This chapter has presented fundamental building blocks of neural networks and deep learning. The chapter begins with introducing neurons and basic feed-forward network architecture, and the forward propagation to compute predictions. It follows by a discussion about network training, from back-propagation, gradient computation, to stochastic gradient decent. A series of neural network types are then introduced, such RNN, LSTM, and CNN. After a brief description of the two major objective functions used in training deterministic neural networks, stochastic neural networks and the Conditional VAE framework are introduced, followed by a section of optimisation using the NVI framework. These neural network modules and its optimisation techniques are the foundation of the methods proposed in this thesis.

Chapter 5

Recurrent Neural Network Language Generators

Recently, results have shown that a machine learning approach with a sufficient amount of training data can outperform pipelined systems with a set of sophisticated rules and domain theories (Deng et al., 2013; Sutskever et al., 2014). The use of RNNs for language generation is motivated by the observation that a trained Recurrent Neural Network Language Model (RNNLM) (Mikolov et al., 2010) is effectively a compact encoding of all of the utterances used to train it. If an RNNLM is made to randomly generate word sequences by sampling the output distribution with the previous word as input, many of the generated sequences will be syntactically correct even if semantically incoherent (Sutskever et al., 2011). Hence, if the RNNLM is conditioned during training by some abstract representation of the required semantics, then it should generate appropriate surface realisations. This is the basis of the Recurrent Neural Network Language Generation (RNNLG) framework shown in Fig. 5.1.

5.1 The Recurrent Language Generation Framework

The input to the generator consists of a dialogue act (DA) comprising a DA-*type* such as *inform*, *request*, *confirm* and a set of zero or more *slot-value* pairs. Thus, the output is an appropriate surface realisation. In Figure 5.1, the input DA is “inform(name=Seven_Days, food=Chinese)" and the output is “Seven Days serves chinese food". To make more efficient use of training data, slot values are *delexicalised* by replacing actual slot values by slot tokens; then the substitution reversed after rendering the output. For any slot, there are five possible slot tokens: *VALUE*, *DONTCARE*, *YES*, *NO*, and *NONE*. Thus, in Figure 5.1, the input to the generator is actually "inform(name=VALUE, food=VALUE)" and the output

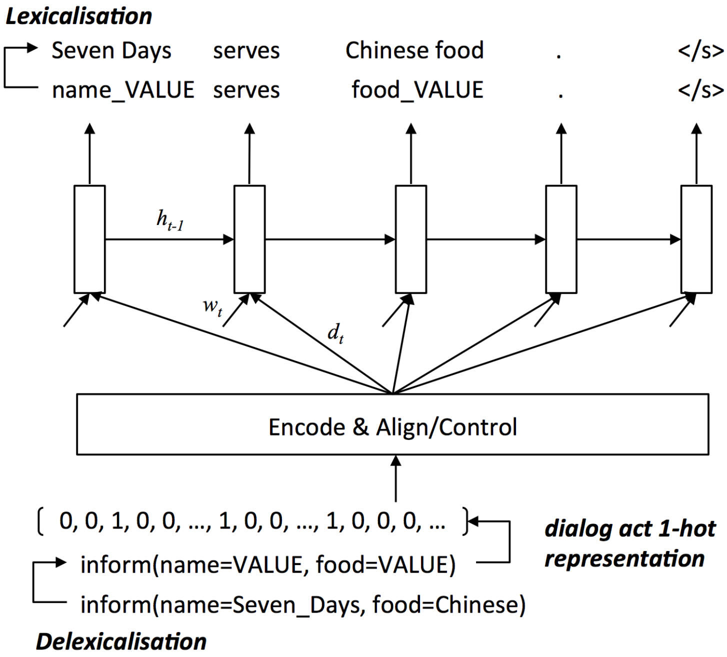

Lexicalisation
Seven Days serves Chinese food . </s>
name_VALUE serves food_VALUE . </s>
h_{t-1}
w_t
d_t
Encode & Align/Control
[0, 0, 1, 0, 0, …, 1, 0, 0, …, 1, 0, 0, 0, …]
dialog act 1-hot representation
inform(name=VALUE, food=VALUE)
inform(name=Seven_Days, food=Chinese)
Delexicalisation

Fig. 5.1 The RNNLG framework for language generation.

is “name_VALUE serves food_VALUE food”. The saved name and food values are then substituted back in as a final post-processing phase.

The framework itself operates as follows. At each time step t, an 1-hot encoding $\mathbf{w}_\mathrm{t}$ of a token[1] w_t is input to the model, which is conditioned on the recurrent hidden layer $\mathbf{h}_{\mathrm{t}-1}$ from the previous step. This entire process produces a new state representation $\mathbf{h}_\mathrm{t}$. To take the system's intended meaning into consideration, the recurrent function f is further conditioned on a control vector $\mathbf{d}_\mathrm{t}$ encoding the system DA,

$$\mathbf{h}_t = f(\mathbf{w}_t, \mathbf{h}_{t-1}, \mathbf{d}_t). \tag{5.1}$$

This updated state is then transformed to an output probability distribution, from which the next token in the sequence is sampled

$$\mathrm{p}(\mathrm{w}_{\mathrm{t}+1}|\mathrm{w}_\mathrm{t}, \mathrm{w}_{\mathrm{t}-1}, ...\mathrm{w}_0, \mathbf{d}_\mathrm{t}) = \mathrm{softmax}(\mathbf{W}_{\mathrm{ho}}\mathbf{h}_\mathrm{t}) \tag{5.2}$$

$$\mathrm{w}_{\mathrm{t}+1} \sim \mathrm{p}(\mathrm{w}_{\mathrm{t}+1}|\mathrm{w}_\mathrm{t}, \mathrm{w}_{\mathrm{t}-1}, ...\mathrm{w}_0, \mathbf{d}_\mathrm{t}). \tag{5.3}$$

Therefore, by decoding input tokens one by one from the output distribution of the RNN until a stop sign is generated (Karpathy and Fei-Fei, 2014), or some other constraint is satisfied

---

[1] We use *token* to emphasise that utterances consist of both words and slot tokens.

(Zhang and Lapata, 2014), the network can produce a sequence of tokens which can be lexicalised[2] to form the required utterance.

As noted in Section 4.1, although there are many choices for the recurrent function f, we opt for an LSTM since it is effective and competitive. These benefits are seen in a variety of configurations based on our preliminary experiments. Therefore, given the input $\mathbf{w}_t$ and previous hidden state $\mathbf{h}_{t-1}$, an LSTM updates its internal state according to the following equations

$$\mathbf{i}_t = \text{sigmoid}(\mathbf{W}_{wi}\mathbf{w}_t + \mathbf{W}_{hi}\mathbf{h}_{t-1}) \tag{5.4}$$
$$\mathbf{f}_t = \text{sigmoid}(\mathbf{W}_{wf}\mathbf{w}_t + \mathbf{W}_{hf}\mathbf{h}_{t-1}) \tag{5.5}$$
$$\mathbf{o}_t = \text{sigmoid}(\mathbf{W}_{wo}\mathbf{w}_t + \mathbf{W}_{ho}\mathbf{h}_{t-1}) \tag{5.6}$$
$$\hat{\mathbf{c}}_t = \tanh(\mathbf{W}_{wc}\mathbf{w}_t + \mathbf{W}_{hc}\mathbf{h}_{t-1}) \tag{5.7}$$
$$\mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \hat{\mathbf{c}}_t \tag{5.8}$$
$$\mathbf{h}_t = \mathbf{o}_t \odot \tanh(\mathbf{c}_t), \tag{5.9}$$

where $\mathbf{i}_t, \mathbf{f}_t, \mathbf{o}_t \in [0,1]^n$ are input, forget, and output gates, respectively, n is the hidden layer size, $\hat{\mathbf{c}}_t$ and $\mathbf{c}_t$ are proposed cell and true cell values, respective, and $\mathbf{W}_{*,*}$ are the model parameters.

The remainder of this section describes three variants of the LSTM. The variants incorporate a control signal to ensure that the LSTM generates an output surface form, which is consistent with the required meaning encoded by the input DA.

5.2 Gating Mechanism

Motivated by the LSTM, which uses gates to control the information flow inside the cell, the same idea can be readily applied to control the input DA; this is called the gating mechanism. Based on the gating mechanism concept, two generators are proposed: (1) the Heuristically Gated LSTM generator (H-LSTM) and (2) the Semantically Conditioned LSTM generator (SC-LSTM).

5.2.1 The Heuristically Gated LSTM Generator

To ensure that the generated utterance represents the intended meaning, a DA control vector $\mathbf{d}_t$ is constructed from the concatenation of 1-hot encodings of the required DA-type and its associated slot-value pairs. The auxiliary information provided by this control vector tends to

---

[2] The process of replacing slot tokens by their saved values.

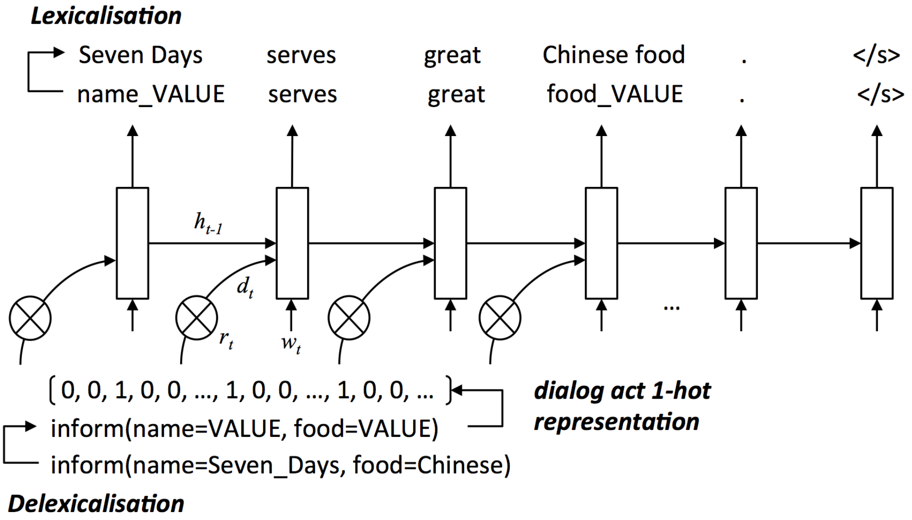


Fig. 5.2 The Heuristically Gated LSTM generator (H-LSTM). At each time step an LSTM cell is used to process the input token and previous time step hidden state. The DA gate is controlled by matching the current generated token with a predefined semantic dictionary.

decay over time because of the *vanishing gradient problem* (Bengio et al., 1994; Mikolov and Zweig, 2012). Hence, $\mathbf{d}_\mathrm{t}$ is reapplied to the LSTM at every time step, as shown in Figure 5.2. Equation 5.8 is then modified to incorporate $\mathbf{d}_\mathrm{t}$

$$\mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \hat{\mathbf{c}}_t + \tanh(\mathbf{W}_{dh}\mathbf{d}_t), \tag{5.10}$$

where $\mathbf{W}_\mathrm{dh}$ is another weight matrix to train.

To prevent the generation of undesirable repetition in the output, the control vector $\mathbf{d}_\mathrm{t}$ is filtered by a reading gate $\mathbf{r}_\mathrm{t}$ at each step before being input into the network

$$\mathbf{d}_t = \mathbf{r}_t \odot \mathbf{d}_{t-1}. \tag{5.11}$$

Each component of this gating signal $\mathbf{r}_\mathrm{t}$ corresponds to a slot-value pair, which is manually reset to zero when the surface form of the slot-value pair appears in the output. To facilitate this, the possible realisations of each slot-value pair must be stored in a semantic dictionary. This model is dubbed the Heuristically Gated LSTM (H-LSTM) (Wen et al., 2015a) generator.

5.2.2 The Semantically Controlled LSTM Generator

The heuristic gating mechanism in the H-LSTM relies on phrase-spotting the generated surface realisation to see when a slot-value has been rendered. Thus, the heuristic gating mechanism is dependent on the provision of a semantic dictionary and exact matching. Cases

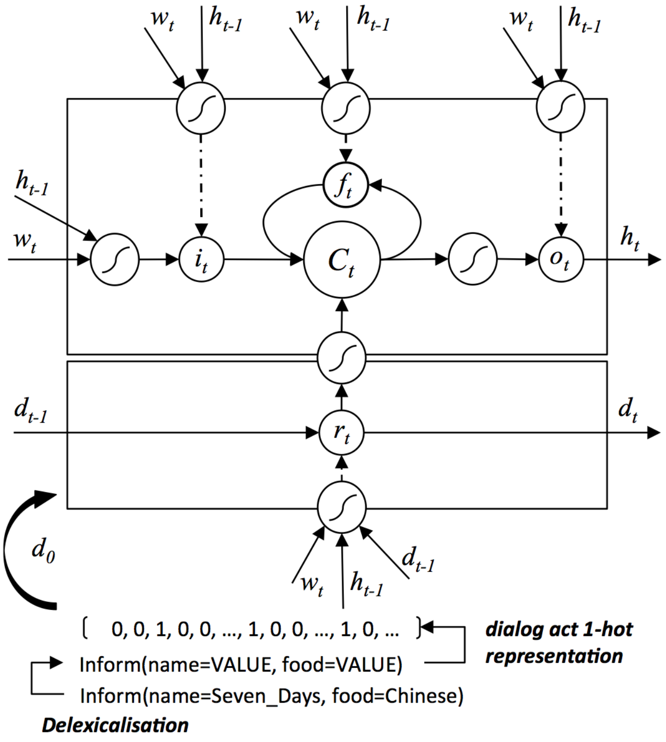


Fig. 5.3 The Semantically Conditioned LSTM generator (SC-LSTM). The upper part is a traditional LSTM cell in charge of surface realisation, while the lower part is a sentence planning cell based on a sigmoid control gate and a DA.

such as binary slots and slots that take "*don't care*" values cannot be explicitly delexicalised in this way; these cases frequently result in generation errors.

One way to make slot-value pair- matching and the corresponding surface forms more flexible is to make the reading gate $\mathbf{r}_t$ trainable by parameterising it using a small network

$$\mathbf{r}_t = \mathrm{sigmoid}(\mathbf{W}_{wr}\mathbf{w}_t + \mathbf{W}_{hr}\mathbf{h}_{t-1} + \mathbf{W}_{dr}\mathbf{d}_{t-1}). \tag{5.12}$$

Here $\mathbf{W}_{wr}$ and $\mathbf{W}_{hr}$ act like keyword and key phrase detectors. The detectors learn to associate certain patterns of generated tokens with certain slot-value pairs. This trainable $\mathbf{r}_t$ function can then be used in Equation 5.11 as a replacement for its heuristic counterpart.

The entire network is trained end-to-end using a cross-entropy cost function, between the predicted word distribution $\mathbf{p}_t$ and the actual word label $\mathbf{y}_t$. The regularisations on the DA control vector and its transition dynamics are characterised by

$$\mathcal{L}(\theta) = \sum_t \mathbf{y}_t^{\mathrm{T}} \log(\mathbf{p}_t) + \|\mathbf{d}_{\mathrm{T}}\| + \sum_{t=0}^{\mathrm{T}-1} \eta \xi^{\|\mathbf{d}_{t+1} - \mathbf{d}_t\|}, \tag{5.13}$$

where θ is the set of all the model parameters, $\mathbf{d}_\mathrm{T}$ is the control vector at the last index T, and η and ξ are constants set to 10^{-4} and 100, respectively. The introduction of the second term is to make sure at the end of the sentence all the slot-value pairs have been rendered into text $||\mathbf{d}_\mathrm{T}|| \to 0$, while the third term is to encourage the model to render one slot-value pair at a time. As shown in Figure 5.3, the Semantically Conditioned LSTM (SC-LSTM) (Wen et al., 2015c) cell is divided into two parts: the lower part is a limited version[3] of *sentence planning*, where it manipulates the control vector features during the generation process to produce a surface realisation that accurately encodes the input information; the upper part is a surface realiser that employs the LSTM to generate coherent utterances.

5.3 Attention Mechanism

Attention mechanisms are very popular in the deep learning community. The core idea of attention is to selectively focus on a certain part of the input patterns when making model decisions. This method has been applied to many NLP tasks such MT (Bahdanau et al., 2015) and image caption generation (Xu et al., 2015). Based on this framework, we propose the Attention-based Encoder Decoder generator (ENC-DEC) to selectively focus on semantic components during generation.

5.3.1 The Attentive Encoder Decoder LSTM Generator

The RNN Encoder-Decoder architecture was first proposed in the MT literature (Bahdanau et al., 2015). The encoder first encodes the input into a distributed vector representation. Then, the decoder subsequently decodes each entry to produce the target output. Adapting an idea from Mei et al. (2015), the encoder uses a separate parameterisation of slots and values. Each slot-value pair is converted into a distributed vector representation $\mathbf{z}_\mathrm{i}$ via the equation

$$\mathbf{z}_\mathrm{i} = \mathbf{s}_\mathrm{i} + \mathbf{v}_\mathrm{i}, \tag{5.14}$$

where $\mathbf{s}_\mathrm{i}$ and $\mathbf{v}_\mathrm{i}$ are the i-th slot and value embedding, respectively, and i runs over the given slot-value pairs. The DA embedding at each time step t is then formed via

$$\mathbf{d}_\mathrm{t} = \mathbf{a} \oplus \textstyle\sum_\mathrm{i} \omega_\mathrm{t,i} \mathbf{z}_\mathrm{i}, \tag{5.15}$$

---

[3] It can only handle cases where each slot-value pair is realised exactly once in the sentence.

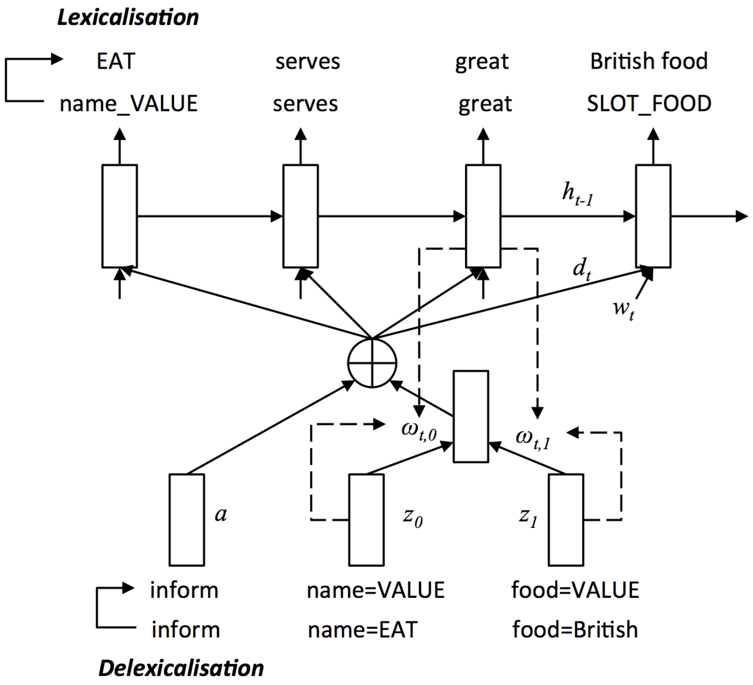


Fig. 5.4 The Attention-based Encoder Decoder generator (ENC-DEC). The DA vector is created by concatenating a deterministic DA embedding with an attentive slot-value embedding over potential slot-value pairs. The dashed lines show how the attentive weights are created by the two control signals.

where **a** is the embedding of the DA type, $\oplus$ is vector concatenation, and $\omega_{\mathrm{t,i}}$ is the weight of i-th slot-value pair calculated by an attention mechanism,

$$\beta_{\mathrm{t,i}} = \mathbf{q}^{\mathsf{T}} \tanh(\mathbf{W}_{\mathrm{hm}}\mathbf{h}_{\mathrm{t}-1} + \mathbf{W}_{\mathrm{mm}}\mathbf{z}_{\mathrm{i}}) \tag{5.16}$$

$$\omega_{\mathrm{t,i}} = \mathrm{e}^{\beta_{\mathrm{t,i}}} / \sum_{\mathrm{i}} \mathrm{e}^{\beta_{\mathrm{t,i}}}, \tag{5.17}$$

where **q**, $\mathbf{W}_{\mathrm{hm}}$, and $\mathbf{W}_{\mathrm{mm}}$ are parameters to train. The DA embedding $\mathbf{d}_{\mathrm{t}}$ is then fed into an LSTM as additional information that will update the hidden state

$$\begin{pmatrix} \mathbf{i}_{\mathrm{t}} \\ \mathbf{f}_{\mathrm{t}} \\ \mathbf{o}_{\mathrm{t}} \\ \hat{\mathbf{c}}_{\mathrm{t}} \end{pmatrix} = \begin{pmatrix} \mathrm{sigmoid} \\ \mathrm{sigmoid} \\ \mathrm{sigmoid} \\ \tanh \end{pmatrix} \mathbf{W}_{4\mathrm{n},3\mathrm{n}} \begin{pmatrix} \mathbf{w}_{\mathrm{t}} \\ \mathbf{h}_{\mathrm{t}-1} \\ \mathbf{d}_{\mathrm{t}} \end{pmatrix}$$

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

$$\mathbf{h}_{\mathrm{t}} = \mathbf{o}_{\mathrm{t}} \odot \tanh(\mathbf{c}_{\mathrm{t}}),$$

Venue	Restaurant	Hotel
informable slots	*pricerange, area, near, food, goodformeal, **kidsallowed**	*pricerange, area, near, **hasinternet**, **acceptscards**, **dogsallowed**
requestable slots	*name, *type, *price, phone, address, postcode	*name, *type, *price, phone, address, postcode
act type	*inform, *inform_only_match, *inform_on_match, *inform_count, *recommend, *select, *confirm, *request, *request_more, *goodbye	
Product	**Laptop**	**Television**
informable slots	*pricerange, family, batteryrating, driverange, weightrange, **isforbusinesscomputing**	*pricerange, family, screensizerange, ecorating, hdmiport, **hasusbport**
requestable slots	*name, *type, *price, warranty, battery, design, dimension, utility, weight, platform, memory, drive, processor	*name, *type, *price, resolution, powerconsumption, accessories, color, screensize, audio
act type	*inform, *inform_only_match, *inform_on_match, *inform_count, *recommend, *select, *confirm, *request, *request_more, *goodbye, inform_all, inform_no_info, compare, suggest	

bold=binary slots, *=overlap between Product and Venue domains, all *informable slots* can take "dontcare" value

Table 5.1 Ontologies of the Restaurant, Hotel, Laptop and TV domains

where n is the hidden layer size, $\mathbf{w}_t$ is the input word embedding, and $\mathbf{W}_{4n,3n}$ are the LSTM parameters.

This kind of attention mechanism can also be viewed as a latent *sentence planning* component where the slot-value pairs $\mathbf{z}_i$ are selected to realise the required output, based on the attention weight $\omega_{t,i}$ at each time step. This is shown in Equation 5.16 and 8.3.

5.4 Data Collection

To collect the data that are required to train our models, the Amazon Mechanical Turk (AMT) service was used. The following section describes the methodology and settings.

5.4.1 Ontologies

To thoroughly evaluate the effectiveness of our methods, experiments were conducted in four different domains: finding a restaurant, finding a hotel, buying a laptop, and buying a television. The ontologies of the four domains are shown in Table 5.1. There are two product domains (Laptop and TV) and two venue domains (Restaurant and Hotel). As we can see, the two product domains are more complicated than the two venue domains because there are more distinct DA-types[4] and slot kinds. More details, e.g., overlapping slots, DAs (marked by * sign) and binary slots that take only binary values (highlighted in **bold**), are presented in Table 5.1. Each DA type represents the intent of the system; this is usually combined with a few domain specific slot-value pairs to form a DA valid for that domain. For example, "*inform(name='alexander b&b', pricerange='cheap', acceptscards='yes')*" is a valid DA for the hotel domain.

5.4.2 Methodology

The corpora for the venue and product domains were collected in two separate data collections (Wen et al., 2015c, 2016b). Both used the Amazon Mechanical Turk (AMT) service. AMT workers were recruited and asked to propose an appropriate natural language realisation corresponding to each system DA presented to them. The set of DAs in the venue domains (Restaurant and Hotel) were generated via an actual dialogue system (Wen et al., 2015c) and each DA could be presented to workers several times. This results in a multiple reference dataset for both the Restaurant and Hotel domains. On the other hand, all possible combinations of DA types and slots were enumerated and only one surface form was collected for each DA (Wen et al., 2016b) in the product domains (Laptop and TV). Therefore, the product domain datasets are harder than the venue domain ones because the product domain datasets have only one reference for each DA while the venue ones contain multiple references. Detail statistics about the collected datasets are presented in Table 5.2.

5.4.3 Dataset Overview

The statistics of the four datasets[5] collected are shown in Table 5.2. Since there are multiple references for each DA in the Restaurant and Hotel domains, the corpora of these two domains are good for evaluating NLG systems using similarity-based metrics like BLEU score (Papineni et al., 2002). On the other hand, since the corpora collected in Laptop and TV domains have a much larger input space, they typically only have one training example for each DA. These corpora are better resources for evaluating the generalisation capability of different methods when faced with unseen semantic inputs. Furthermore, due to the varying characteristics of the venue and product domains, the ability to adapt well between the two is

---

[4] One thing worth to note is the *compare* DA, which prompts the system to generate Sparky-like value-based comparisons.

[5] Datasets are available together with the code at https://github.com/shawnwun/RNNLG

Property	Restaurant	Hotel	Laptop	Television
# of unique DAs	248	164	∼13K	∼7K
# of data points	∼5K	∼5K	∼13K	∼7K
# of ref. per unique DA	multiple	multiple	single	single

Table 5.2 Properties of the four datasets collected

a good indicator of the effectiveness of the adaption method under test. We will investigate the adaptation of RNNLG in Chapter 6.

5.5 Baselines

To assess the performance of the proposed RNNLG models, we compared them against three non-NN based methods: (1) a handcrafted, template-based generator as described in Section 2.5.1, (2) an example-based generator based on k-Nearest Neighbour (kNN) as mentioned in Section 3.3, and (3) a class-based LM generator introduced in Section 3.1. Details are provided below:

Handcrafted Generator A template-based generator (*hdc*) developed in the Cambridge Dialogue Systems Group is the basis for the handcrafted generator used in this thesis. It employs a lookup-table for each configuration of the input DA to extract a surface template containing a set of slot-value related placeholders. Based on realistic slots and values in the DA, the generator recursively realises parts of the surface form via additional table lookups. The handcrafted generator was tuned over a long period and has been used frequently to interact with real users (Gasic and Young, 2014). Therefore, it sets a reasonable bar even though it is not the optimal baseline.

K-Nearest Neighbour-based Generator The k-Nearest Neighbour (*knn*) approach is an example-based method for NLG as described in Section 3.3. During training, all the sentences are delexicalised with their corresponding DA represented in one-hot feature vectors. The DA feature vectors were used to computed the cosine similarities between testing DAs and all DAs in the training set to find the highest scored template for realisation. The template used for final realisation is sampled if multiple templates are ranked equally.

Class-based Language Model The third baseline approach we compared to is the class-based LM (*ngram*) generator (Oh and Rudnicky, 2000), as introduced in Section 3.1. Due to the inefficiency of the class-based LM generator, previous studies (Mairesse and Young, 2014; Wen et al., 2015a) have shown that clustering sentences into finer classes while using a longer n-gram can help the model perform better. Consequently, we experimentally compared our models to a class-based LM that has a n-gram length of 5 and divides its classes by the DA type and up to 3 slots (compared to only 1 slot in the original paper). During decoding, we also allowed the n-gram models to over-generate 10 times more sentences than our RNNLG models (200 v.s. 20).

Other baselines This thesis mainly compares the proposed approaches with corpus-based baselines so that different models can be trained under the same settings with the same data. Approaches like Phrase-based DBN (Section 3.2), SPaRKy or other linguistically-motivated methods (Section 2.5.2) are not considered because they require additional information apart from the input-output examples such as tree structure rules and semantic alignments. Note both Mei et al. (2015) and Dušek and Jurcicek (2016) were also excluded from the experiments here because they are essentially attention-based encoder-decoder neural network architecture where the proposed ENC-DEC network can well represent their performance.

5.6 Corpus-based Evaluation

5.6.1 Experimental Setup and Evaluation Metrics

The LSTM-based natural language generators were implemented using the Theano library (Bastien et al., 2012; Bergstra et al., 2010)[6]; they were trained by partitioning each of the collected corpora into a training, validation and testing were set in the ratio 3:1:1. All the generators were trained by treating each sentence as a mini-batch. The parameters were randomly initialised between -0.3 and 0.3 and an l_2 regularisation term was added to the objective function every 10 training examples. The hidden layer size was set to 80 for all cases. Stochastic gradient descent and back propagation through time (Werbos, 1990) were used to optimise the parameters. To prevent overfitting, early stopping was implemented using the validation set. To decode utterances from a trained model, an over-generation and re-ranking approach was adopted Oh and Rudnicky (2000). Firstly, 20 utterances were generated using beam search with the beam-width set to 10, where the decoding criteria is the average log-likelihood of the utterance. The top 5 realisations were then selected for each DA according to the following re-ranking criteria,

$$\mathrm{R} = -(\mathcal{L}(\theta) + \lambda \cdot \mathrm{ERR}) \tag{5.18}$$

---

[6] Code available at https://github.com/shawnwun/RNNLG

where λ is a trade-off constant, $\mathcal{L}(\theta)$ is the cost generated by a model with parameters θ, and the slot error rate ERR is computed by exact matching of the slot tokens in the candidate utterances as described in Equation 3.14, Section 3.5.2. λ is set to a large value (10) to severely penalise nonsensical outputs.

The various models were tested using both corpus-based evaluation and human evaluation. In the corpus-based evaluations, model performance was assessed using two objective evaluation metrics as noted in Section 3.5, the BLEU score (Papineni et al., 2002) and slot error rate ERR (Wen et al., 2015c) as described in Equation 3.14, Section 3.5.2. Both metrics were computed based on the top 5 realisations. Slot error rates were calculated by averaging slot errors over each of the top 5 realisations in the entire corpus. Multiple references were used to compute the BLEU scores where available (i.e. for the restaurant and hotel domains), but for corpora that contain only single references are only computed based on the single reference[7]. Since the generators are stochastic and the trained networks can differ depending on the initialisation, the corpus-based evaluation results shown in Table 5.3 were produced by training each neural network model on 5 different random seeds (1-5) and selecting the models with the best BLEU score on the validation set.

5.6.2 Results

The proposed RNNLG methods described in Section 5.1 (i.e., the Heuristically Gated LSTM (*hlstm*), the Semantically Conditioned LSTM (*sclstm*), and the Attentive Encoder-Decoder (*encdec*)) were first compared with the baseline models in terms of generation quality on the held-out test sets. The results are shown in Table 5.3.

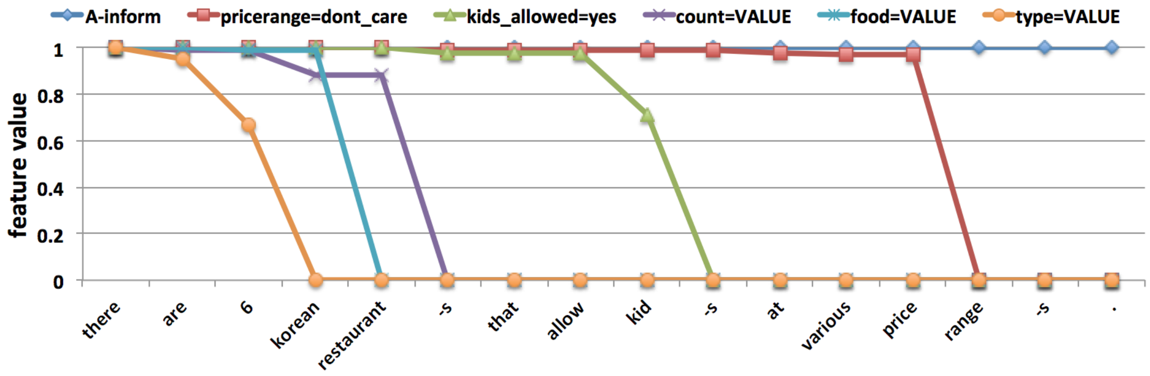
The rule-based baseline (*hdc*) performed the worst, in terms of BLEU score, across all four domains. Nevertheless, it achieved zero slot error rates. Setting aside the difficulty of scaling to large domains, the handcrafted generator's use of predefined rules yields a fixed set of sentence plans, which can differ markedly from the colloquial human responses collected using AMT. The class-based language model approach (*ngram*), on the other hand, generally suffers from inaccurate rendering of information, which results in very poor slot error rates across the four domains. However, due to language modelling, the *ngram* model can achieve a better BLEU score than the *hdc* baseline. The kNN model (*knn*), although not optimal, performed robustly across four domains on both metrics. Since *knn* is an example-based method that picks its realisation template directly from the training examples, it cannot go too wrong except for the DAs that don't exist in the training set (poor generalisation).

---

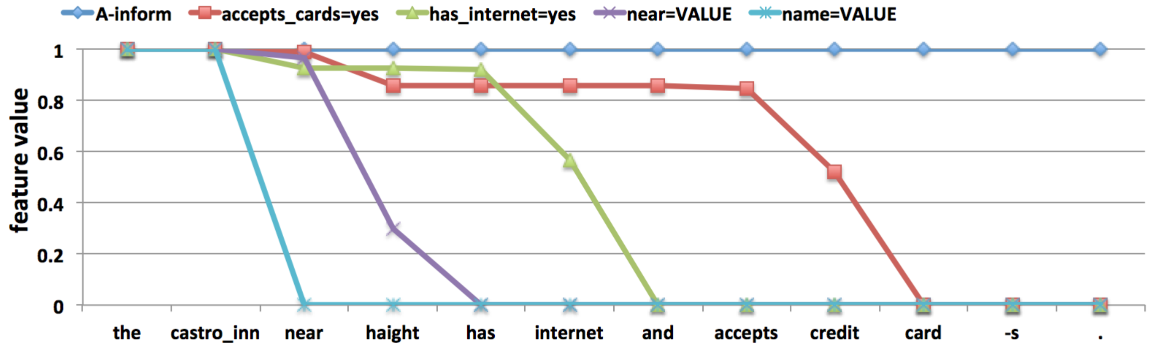
[7] This may not be a reliable metric for more complex scenarios but we found that for the particular task discussed in this thesis it is good enough.

Domain	Model	test set		validation set	
		BLEU	ERR(%)	BLEU	ERR(%)
Restaurant	hdc	0.4260	0.00	-	-
	knn	0.594	0.60	-	-
	ngram	0.642	8.73	-	-
	encdec	0.740	2.78	0.775	2.75
	hlstm	0.747	0.74	0.779	0.73
	sclstm	**0.753**	**0.38**	**0.786**	**0.47**
Hotel	hdc	0.5406	0.00	-	-
	knn	0.675	**1.75**	-	-
	ngram	0.770	5.87	-	-
	encdec	**0.855**	4.69	**0.852**	4.56
	hlstm	0.850	2.67	0.849	**2.14**
	sclstm	0.848	3.07	0.851	2.38
Laptop	hdc	0.3761	0.00	-	-
	knn	0.414	0.88	-	-
	ngram	0.294	35.64	-	-
	encdec	0.511	4.04	**0.517**	3.83
	hlstm	**0.513**	1.10	0.512	1.42
	sclstm	0.512	**0.79**	0.516	**0.95**
TV	hdc	0.3919	0.00	-	-
	knn	0.432	3.69	-	-
	ngram	0.298	38.78	-	-
	encdec	0.518	3.18	0.520	3.49
	hlstm	0.525	2.50	0.527	2.17
	sclstm	**0.527**	**2.31**	**0.527**	**2.02**

Table 5.3 Corpus-based evaluation on four domains. The results were produced by training each model on 5 random seeds (1-5) and selecting the models with the best BLEU score on the validation set. Results of both the test and the validation set are reported. The approach with the best performance on a metric in a domain is highlighted in **bold**, while the worst is underlined.



(a) An example realisation from the Restaurant domain



(b) An example realisation from the Hotel domain

Fig. 5.5 Examples showing how the SC-LSTM controls the dialogue features, flowing into the network through the reading gates. Despite errors due to sparse training data for some slots, each gate generally learned to detect the words and phrases relating to each associated slot-value pair.

Comparing the three baseline methods (*hdc, knn, ngram*) with the three RNNLG models (*encdec, hlstm, sclstm*), the RNNLG models consistently outperformed the baselines in terms of BLEU score. This is presumably because of the core LSTM efficiency learning the natural language features embedded in the training set and thereby reproducing more natural utterances. Although the BLEU scores do not differ too much between the RNNLG models, the different architectures used to handle the content selection and planning do have a strong effect on the slot error rate. Overall, the gating mechanisms implemented by H-LSTM and SC-LSTM are more effective than the attention mechanism used in ENC-DEC for reducing the slot error rate. This is probably because the former gating mechanism can effectively prevent the undesirable repetition of semantic elements by turning the slot off permanently, once the information for that slot has been rendered whereas the softer attention mechanism can still allow repetition.

#	Example Dialogue Acts and Realizations from Restaurant Domain
1	inform(name="red door cafe", goodformeal="breakfast", area="cathedral hill", kidsallowed="no") *red door cafe is a good restaurant for breakfast in the area of cathedral hill and does not allow children .* *red door cafe is a good restaurant for breakfast in the cathedral hill area and does not allow children .* *red door cafe is a good restaurant for breakfast in the cathedral hill area and does not allow kids .* *red door cafe is good for breakfast and is in the area of cathedral hill and does not allow children .* *red door cafe does not allow kids and is in the cathedral hill area and is good for breakfast .*
2	informonly(name="dosa on fillmore and kiss seafood", pricerange="expensive", near="lower pacific heights") *there is no place other than dosa on fillmore and kiss seafood that are expensive near to lower pacific heights .* *dosa on fillmore and kiss seafood is the only expensive restaurant near lower pacific heights .* *the only listed restaurant near lower pacific heights in the expensive price range is dosa on fillmore and kiss seafood .* *i apologize , dosa on fillmore and kiss seafood is the only expensive restaurant near lower pacific heights .* *i apologize , dosa on fillmore and kiss seafood are the only expensive restaurants near lower pacific heights .*
#	**Example Dialogue Acts and Realizations from Hotel Domain**
3	inform(type="hotel",count="182",dogsallowed="dontcare") *there are 182 hotels if you do not care whether dogs are allowed .* *there are 182 hotels if you do not care whether they allow dogs .* *182 hotels are available if dogs allowed or not is not an issue .* *there are 182 hotels if allowing dogs or not is not an issue .* *there are 182 hotels if whether dogs are allowed does not matter .*
4	informonly(name="red victorian bed breakfast",acceptscreditcards="yes",near="haight",hasinternet="yes") *red victorian bed breakfast is the only hotel near haight and accepts credit cards and has internet .* *red victorian bed breakfast is the only hotel near haight and has internet and accepts credit cards .* *red victorian bed breakfast is the only hotel near haight that accept credit cards and offers internet .* *the red victorian bed breakfast has internet and near haight , it does accept credit cards .* *the red victorian bed breakfast is the only hotel near haight that accepts credit cards , and offers internet .*

Table 5.4 Samples of top 5 realisations from the SC-LSTM output.

Among the four domains tested, the SC-LSTM performs best in most of cases, especially in the Restaurant and TV domains. Its ability to learn words or phrases that trigger the semantic gate is key. Examples showing how the SC-LSTM controls the reading gates can be seen in Figure 5.5.

Table 5.4 shows the examples of the output generated from SC-LSTM in two venue domains. Despite the fact that most of sentences are correct both semantically and syntactically, there are still some grammatical errors from time to time. For example, the "accept" in the third sentence of the fourth block should be "accepts" because it is present simple tense.

5.7 Human Evaluation

5.7.1 Experimental Setup and Evaluation Metrics

To confirm the findings of the corpus-based evaluation in Table 5.3, a human evaluation was conducted in the Restaurant and Laptop domains and the results are shown in Tables 5.5 and 5.6, respectively. Two tests were run in each domain: (1) an utterance quality test in which AMT workers were asked to rate each utterance from 1 to 5 in terms of how informative and natural each is, and (2) a pairwise preference test, where the workers were asked to compare two models w.r.t. an associated test example and state their preference.

Utterance Quality Evaluation (Rating ranges from 1 to 5)

Metrics	hdc	ngram	knn	hlstm	sclstm	encdec	ref
Informativeness	3.88	3.60^{**}	3.91	4.00	3.99	3.95	3.93
Naturalness	3.46^{**}	3.34^{**}	3.69	3.73	3.80	3.70	3.81

Pairwise Preference Comparison (%)

	hdc	ngram	knn	hlstm	sclstm	encdec	ref
hdc	-	35.5	25.5	26.8	42.6	33.9	38.6
ngram	64.5^{*}	-	37.3	26.6	22.4	35.4	30.9
knn	74.5^{**}	62.7	-	38.6	35.2	42.6	36.4
hlstm	73.2^{**}	73.4^{**}	61.4	-	43.8	50.8	55.3
sclstm	57.4	77.6^{**}	64.8^{*}	56.2	-	59.6	43.4
encdec	66.1^{*}	64.6	57.4	49.2	40.4	-	61.2
ref	61.4	69.1^{*}	63.6	44.7	56.6	38.8	-

* $p < 0.05$, ** $p < 0.005$

Table 5.5 Human evaluation of the Restaurant domain. The significance of the *Utterance Quality Evaluation* section indicates whether a model is *significantly worse* than the reference based on a Wilcoxon signed-rank test. The *Pairwise Preference Comparison* section indicates whether a model is *significantly preferred* to its opponent based on a two-tailed binomial test.

Here *informativeness*[8] is defined as an utterance that contains all the information specified in the DA. The term *naturalness*[9] is defined as an utterance that has been produced by a human. The significance flag in the *Utterance Quality Evaluation* section indicates whether a model is *significantly worse* than the reference. In the *Pairwise Preference Comparison* section, the evaluation indicates whether a model is *significantly preferred* to its opponent.

5.7.2 Results

The results for the Restaurant domain are shown in Table 5.5. The ratings of the reference (*ref*) are only significantly better than the handcrafted (*hdc*) and class-based language model (*ngram*) baselines but not the rest. This indicates that the RNNLG models are considered indistinguishable from the human authored sentences in this domain. This mainly comes from the ability of the LSTM to model long range dependencies and its powerful generalisation

---

[8] The exact wording:"*A higher informativeness score means the computer conveys exactly the same idea as in the reference sentence.*"

[9] The exact wording:"*A higher naturalness score means the computer behaves more like a human.*"

Utterance Quality Evaluation (Rating ranges from 1 to 5)							
Metrics	hdc	ngram	knn	hlstm	sclstm	encdec	ref
Informativeness	4.14	2.84**	4.04**	3.99**	4.00**	3.97**	4.28
Naturalness	3.51**	2.87**	3.64*	3.73	3.71	3.60**	3.84

Pairwise Preference Comparison (%)							
	hdc	ngram	knn	hlstm	sclstm	encdec	ref
hdc	-	86.8**	36.1	16.7	23.5	30.0	36.1
ngram	13.2	-	24.0	14.3	13.8	12.5	11.1
knn	63.9	76.0*	-	40.5	40.5	31.0	60.5
hlstm	83.3**	85.7**	59.5	-	51.9	50.8	48.0
sclstm	76.5**	86.2**	59.5	48.1	-	57.7	59.3
encdec	70.0*	87.5**	69.0	49.2	42.3	-	50.0
ref	63.9	88.9**	39.5	52.0	40.7	50.0	-

* $p < 0.05$, ** $p < 0.005$

Table 5.6 Human evaluation of the Laptop domain. The significance flag in the *Utterance Quality Evaluation* section indicates whether a model is *significantly worse* than the reference based on a Wilcoxon signed-rank test, while in the *Pairwise Preference Comparison* section it indicates whether a model is *significantly preferred* to its opponent based on a two-tailed binomial test.

capability over unseen inputs. Among the approaches evaluated, the SC-LSTM (*sclstm*) seems to be the best choice when both metrics are considered, although the differences are not statistically significant. In the preference test, the judges preferred the SC-LSTM to all the other methods. These observations are consistent with the corpus-based evaluation that SC-LSTM seems to be the best choice for this task. Furthermore, these results suggest that the gating mechanism is more effective than the attention mechanism since the H-LSTM and SC-LSTM were both rated higher for the two quality scores and preferred by the judges to the ENC-DEC model.

The results in the Laptop domain in Table 5.6 are less clear cut. As in the Restaurant domain, the gating mechanism is better than the attention mechanism. However, in this case, the H-LSTM seems to be marginally preferred than the SC-LSTM. This is probably due to the complexity and difficulty of the Laptop domain. This results in the semantic control gates of the SC-LSTM being undertrained. In this case, the predefined heuristics of the H-LSTM provides a more robust control mechanism.

5.8 Conclusions

This chapter has presented a thorough study that compares the proposed RNNLG models with three baseline methods and against each other. The template-based approach is the most robust, in terms of slot error rate, simply because the rendering of information is strictly managed by template rules. Such a deterministic process sacrifices the improvisation in human language; it also generates surface forms that are much more rigid and stylised than the other approaches. At the other end of the spectrum, the class-based LM sample words produce a lot of variation in the surface forms. Its inefficient use of the DA information often leads to unnecessary computation overhead and erroneous outputs. Despite the poor generalisation over unseen inputs, the kNN-based approach, which uses a similarity measure between input DAs to select templates for realisation, is the most reliable baseline method among the three.

When comparing the three proposed RNNLG models (e.g., H-LSTM, SC-LSTM, ENC-DEC) to the baselines, the following conclusions can be drawn from the results

1. The RNNLG models all outperform baselines in both corpus-based and human evaluations. This can be attributed to the distributed representation and the modelling of long-term dependencies using LSTM.
2. The proposed gating mechanism is preferred over the popular attention mechanism, especially in terms of the slot error rate metric. This is because the gating mechanism is more efficient at preventing semantic repetitions via its hard decision-gates.
3. The SC-LSTM is the top choice for most of the domains due to its learnable controlling gates. However, H-LSTM can be comparable in complex domains, where the input space is rich (more slots, values and DA types) and the amount of data is relatively insufficient (single reference per DA).
4. The on and off of SC-LSTM's internal gates can be interpreted as the acquired semantic alignments between the semantic components (slots and values) in the meaning representation and the partial realisation of the surface form.

Therefore, the proposed SC-LSTM can be considered a more suitable model for the tasks. A study of its domain scalability will be presented in the next chapter.

Chapter 6

Domain Adaptation

A crucial advantage of a statistical-based NLG system is the ability to scale it to cover more domains by simply collecting more data, without the need for specific domain knowledge or design know-how. However, data collection itself can become costly and laborious if there are many new domains. *Domain adaptation* provides a solution to this problem. Domain adaptation problems arise when we have sufficiently labelled data in one domain (the *source* domain $\mathbb{S}$), but have little or no labelled data in a related domain (the *target* domain $\mathbb{T}$). The goal is to design a learning algorithm that can use large source domain data sets to facilitate learning and prediction in the target domain. Whether the target domain data is labelled or not, we can roughly divide the problem into two categories: (1) the fully supervised case where a large amount of source domain data and a small labelled dataset in the target domain are available; (2) the semi-supervised case where the source domain dataset is annotated but the target domain dataset is not. In this thesis, we exclusively focus on the fully supervised case because it suits the NLG scenario better.

From a machine learning perspective, where the type of adaptation applied to the pipeline matters, this yields three major methods for domain adaptation: data augmentation, feature-based domain adaptation, and model-based domain adaptation.

6.1 Data Augmentation

Data augmentation (Frühwirth-Schnatter, 1994; Tanner and Wong, 1987) is a method that is widely applied in machine learning; the method can create a rich dataset from a small set of authentic data points. Although it may not necessarily solve the data sparsity problem, it does improve the model's robustness. For example, if an image contains a cat, then a proper scale, rotation, shift or change of colour of that image shouldn't affect this fact. Therefore, to perform well on these variants, the model should be trained on a set of pictures properly transformed from the original image rather than just itself.

Data augmentation is popular in the Computer Vision (CV) community and is often combined with Deep Neural Networks to prevent overfitting (Chatfield et al., 2014; Krizhevsky et al., 2012). This technique has also been applied to document classification (Zhang and LeCun, 2015) in NLP to augment text data via a thesaurus. Relation extraction in biomedical NLP uses data augmentation to create new instances by modifying the headword candidate arguments by covering equivalent arguments. Moreover, data augmentation was also used to generate negative examples (Bordes et al., 2015) during training when the dataset contains mostly positive examples; obtaining the negative ones is not trivial. Although data augmentation-based domain adaptation has not been studied extensively in the past, its application to NLG is intuitive and shows impressive results. The most relevant work was done by Hogan et al. (2008). They showed that an LFG f-structure-based generator could yield better performance if trained on in-domain sentences paired with pseudo-parse tree inputs generated from a state-of-the-art, but out-of-domain parser. Furthermore, Cuayahuitl et al. (2014) trained statistical surface realisers from unlabelled data via an automatic slot labelling technique. Our approach employs a data augmentation-type approach. More details can be found in Section 6.5.1.

6.2 Feature-based Domain Adaptation

Unlike data augmentation, which directly modifies the input data, feature-based domain adaptation relies on manipulating the feature space to adapt models across domains. There are roughly three different families (classes) of this approach.

Feature augmentation is the method proposed by Daume III (2007). This method duplicates the original feature vector three times. Each replicated feature vector represents general, source-specific, and target-specific features. Therefore, the augmented source data contains only the general and source-specific features, while the augmented target data contains only the general and target-specific versions. The model can then learn to put weights on the general features that are shared across domains, but also distinguish features that are domain-specific by separating the source and target specific feature space. An updated kernel function $\hat{\mathrm{K}}(\mathrm{x},\mathrm{x}')$ is then treated as a measure of similarity between a pair of data points and applied to the augmented data points via

$$\hat{K}(x,x') = \begin{cases} 2\mathrm{K}(\mathrm{x},\mathrm{x}'), & \text{if same domain,} \\ \mathrm{K}(\mathrm{x},\mathrm{x}'), & \text{if different domain,} \end{cases} \tag{6.1}$$

where x and x' are data points and $\mathrm{K}(\mathrm{x}, \mathrm{x}')$ is the original kernel function. Note Equation 6.1 is self-explanatory and indicates that the data points from the target domain have twice as much influence as the source data points when making decisions about testing data from the target domain. Based on this method, many algorithms such as SVM and maximum entropy models can be used to optimise parameters.

Feature ensemble (Xia et al., 2013) is another feature-based adaptation technique that learns the ensemble weights based on a set of pre-trained models. The approach begins by dividing the feature vector **x** into sub-groups k. For each sub-group, a classifier $\mathrm{g}_\mathrm{k}(\mathbf{x}_\mathrm{k})$ is trained on those sub-group features only

$$g_k(\mathbf{x}_k) = \omega_k^\intercal \mathbf{x}_k, \tag{6.2}$$

here a linear classifier is used as the base classifier for each sub-group. After base classification, a meta-learning approach is stacked on top to learn the ensemble weight of each base classifier θ_k based on the small amount of annotated data from the target domain

$$f(\mathbf{x}) = \sum_k \theta_k g_k(\mathbf{x}_k). \tag{6.3}$$

Note this approach differs from the model ensemble described in Section 6.3. In this case, each base classifier is only trained on a subset of the original feature space. The model ensemble technique combines models trained on a subset of data points in the full feature space. In the original work, the feature ensemble was also combined with a sample selection that selects a subset of data points in the source domain whose instance distribution is close to the target domain that trains the model.

The feature embedding approach becomes more and more popular because of the rise in popularity of deep learning, even though it has existed for a while (Blitzer et al., 2006; Yang and Eisenstein, 2015). This family of approaches aims at finding a mapping that maps both the source domain data points and the target domain data points into a shared feature space on which the classifier is trained on. Therefore, if a good mapping is developed, then the classifier trained in the source domain will also be effective in the target domain. Previously, this feature mapping was learnt separately, based on Singular Value Decomposition (SVD) as in Blitzer et al. (2006). This mapping has recently become an NN-based embedding function that has become common practise (Collobert and Weston, 2008; Yang and Eisenstein, 2015).

6.3 Model-based Domain Adaptation

Model-based domain adaptation is probably the most popular approach among the three methods because it is much more intuitive. It covers a wide range of methods such as modifying the learning signals (Lemon, 2008; Walker et al., 2007), changing the model architecture (Mrkšić et al., 2015; Wen et al., 2013a), model ensemble (Heidel and Lee, 2007; Wen et al., 2012a), and learning a transformation for model parameters (Gales and Woodland, 1996; Leggetter and Woodland, 1995). Most of the current speech and language applications rely on adaptation of this method because the method doesn't always require the model to be re-trained , which allows online domain adaptation.

The most direct approach to model-based domain adaptation is to continue to train the model based on data points from the target domain. This often requires a change in the model architecture or specific steps during training so that the adapted model can perform reasonably well in the target domain. For example, Mrkšić et al. (2015) trained a general belief tracker by abstracting away the identity of slots. This procedure was later adapted to track each individual slot. Wen et al. (2013a) personalised an RNN LM by adapting a model trained on a general corpus to specific social network users. The RNN they used was augmented with an additional feature layer to facilitate information sharing between different users (or domains). In this thesis, the proposed adaptation recipe is compared with a model-based adaptation baseline called model fine-tuning. A general model is fine-tuned by being trained on the source domain data and then further adapted by tuning the parameters based on the adaptation data. More details can be found in Section 6.4.

Model ensemble is a domain adaptation technique that is especially popular for language modelling. The idea of model ensemble is similar to a feature ensemble as described in Section 6.2. However, instead of training models based on a subset of features, model ensemble learns models based on a subset of training examples and combines them later by learning a set of model-dependent weights. In language modelling (Heidel and Lee, 2007; Wen et al., 2012a), for example, a set of class-based LMs is first trained on topics like politics, tourism, sports, or economics. During testing, the inference algorithm will dynamically decide the weight of each topic LM and combine them into a single LM, which can be used to estimate the complexity of the test data.

Most of the adaptation work done in NLG were model-based methods. However, most of them did not directly address the problem of domain adaptation. The SPoT-based generator, proposed by Walker et al. (2002), has the potential to address domain adaptation problems. However, their published work has focused on tailoring user preferences (Walker et al., 2007) and mimicking personality traits (Mairesse and Walker, 2011). In contrast, Mairesse et al.

(2010) proposed the use of active learning to mitigate the data sparsity problem when training data-driven NLG systems.

6.4 Baseline Method - Model Fine-Tuning

Given training instances represented by DA $\mathrm{d_i}$ and sentence $\mathrm{y_i} = \{\mathrm{w_0^i}, \mathrm{w_1^i}, ...\mathrm{w_N^i}\}$ tuples $\{\mathrm{d_i}, \mathrm{y_i}\}$ from the source domain $\mathbb{S}$ (data rich) and the target domain $\mathbb{T}$ (data sparse), our goal is to find a set of network parameters $\theta_{\mathbb{T}}$ that can perform acceptably well in the target domain. A straightforward way to adapt NN-based models to a target domain is to continue training, or fine-tuning, a well-trained generator on whatever new target domain data is available. This belongs to one of the model-based approaches described in the previous section. The training procedure is as follows:

1. Train a source domain generator $\theta_{\mathbb{S}}$ on source domain data $\{\mathrm{d_i}, \mathrm{y_i}\} \in \mathbb{S}$ with all values delexicalised[1].
2. Set $\theta_{\mathbb{T}} = \theta_{\mathbb{S}}$ as the initial parameters for the target domain.
3. Divide the adaptation data into training and validation sets. Refine parameters by training on adaptation data $\{\mathrm{d_i}, \mathrm{y_i}\} \in \mathbb{T}$ with early stopping and a smaller starting learning rate. This yields the target domain generator $\theta_{\mathbb{T}}$.

This method is very like the work of Wen et al. (2013a), where the adaptation was done by continuing to train the source domain model on target domain data. Although this method can benefit from parameter sharing part of the LM network, the parameters of similar input slot-value pairs are not shared[1]. Thus, the realisation of any unseen slot-value pairs in the target domain can only be learned from scratch and adaptation offers no benefit in this case.

6.5 The Adaptation Recipe for Language Generators

The feature-based method requires manual manipulation of the input feature space; therefore, it is not very suitable for combining with NN-based approaches since the power of NN mainly comes from the ability of learning from raw features. Moreover, model-based approaches are tricky to implement in practise for slot-value based applications. They are either like the

---

[1] We have tried training with both delexicalised slots and values and then use the weights to initialise unseen slot-value pairs in the target domain. However, this process yielded even worse results since the learned semantic alignment became stuck at a local minimum. Pre-training only the LM parameters did not produce better results.

method mentioned in Section 6.4, which does not make the best use of the source domain data or require a complex weight-typing strategy and pre-training like Mrkšić et al. (2015), which is more difficult to implement for NLG tasks. On the other hand, data augmentation suits the problem quite well since the RNNLG models already require delexicalisation as the pre-processing step. Consequently, the adaptation recipe proposed in this thesis is centred on the idea of data augmentation and the more efficient use of the data at hand. This leads to the combination of two methods: Data Counterfeiting and Discriminative Training, as described in the following two sections.

6.5.1 Data Counterfeiting

To maximise the effect of domain adaptation, the model should be able to (1) generate acceptable realisations of unseen slot-value pairs (based on similar slot-value pairs seen in the training data), and (2) continue to distinguish slot-value pairs that are similar but nevertheless distinct. Instead of exploring weight-tying strategies in different training stages (which is difficult to implement and typically relies on ad-hoc rules), we propose instead a data counterfeiting approach to *synthesise* target domain data from source domain data. The procedure is shown in Figure 6.1 and described in the list below:

1. Categorise slots, both source and target domain, into classes according to some similarity measure. In our case, we categorise the slots based on their functional type to yield three classes: *informable (I)*, *requestable (R)*, and *binary (B)*[2]. For more details about the slots in the three classes, please refer back to Table 5.1.
2. Delexicalise all slots and values to produce template-like sentences.
3. For each slot s in the source example $(\mathrm{d_i}, \mathrm{y_i})$, randomly select a new slot $\mathrm{s'}$ to replace s. Note that $\mathrm{s'}$ must belong to the target domain and in the same functional class as s. This yields a pseudo example $(\mathrm{d'_i}, \mathrm{y'_i}) \in \mathbb{T}$ in the target domain.
4. Train a generator $\theta'_{\mathbb{T}}$ on the counterfeit dataset $\{\mathrm{d'_i}, \mathrm{y'_i}\} \in \mathbb{T}$.
5. Refine the parameters on in-domain data. This yields the target domain model $\theta_{\mathbb{T}}$.

The name counterfeiting comes from the fact that the training examples used to train the target domain generator are not authentic data points but counterfeited by sampling slots from the target to replace the original ones. This approach allows the generator to share realisations

---

[2] The *informable* class includes all non-binary informable slots while the *binary* class includes all binary informable slots.

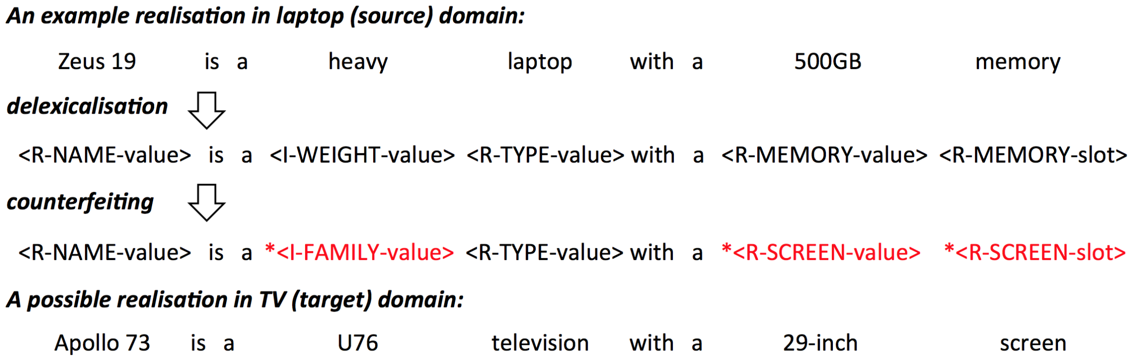

An example realisation in laptop (source) domain:
Zeus 19 is a heavy laptop with a 500GB memory
delexicalisation
<R-NAME-value> is a <I-WEIGHT-value> <R-TYPE-value> with a <R-MEMORY-value> <R-MEMORY-slot>
counterfeiting
<R-NAME-value> is a *<I-FAMILY-value> <R-TYPE-value> with a *<R-SCREEN-value> *<R-SCREEN-slot>
A possible realisation in TV (target) domain:
Apollo 73 is a U76 television with a 29-inch screen

Fig. 6.1 An example of the data counterfeiting approach. Note the slots highlighted *red* are randomly sampled from the target domain where the prerequisite is that the counterfeited slot needs to be in the same functional class as the original slot.

among the slot-value pairs that have similar functionalities. Therefore, it facilitates the transfer learning of rare slot-value pairs in the target domain. Furthermore, the approach also preserves the co-occurrence statistics of slot-value pairs and their realisations. This allows the model to learn a gating or attention mechanism even before adaptation data is introduced.

6.5.2 Discriminative Training for Adaptation

Apart from data counterfeiting, the second approach in the proposed adaptation recipe is Discriminative Training (DT). DT was an alternative objective function to MLE when training ASR or language models (Collins, 2002; Kuo et al., 2002) to improve model predictions. Instead of maximising the log-likelihood of the correct examples, its core idea is to separate the correct examples from competing incorrect ones.

Based on the notations introduced in Section 4.2.2, we have the current model parameterised using θ and training examples $(\mathrm{d_i},\mathrm{y_i})$ where $\mathrm{d_i}$ is the input DA and $\mathrm{y_i}$ is the reference sentence. Following Equation 4.26, the DT objective function is therefore written as,

$$\theta_{DT} = \underset{\theta}{\operatorname{argmax}} \sum_{\hat{y}_i \in GEN(d_i)} p_\theta(\hat{y}_i|d_i) R(\hat{y}_i, y_i) \tag{6.4}$$

where $\hat{\mathrm{y}}_\mathrm{i} \in \mathrm{GEN}(\mathrm{d_i})$ is a set of candidate examples generated by the current model. To consider DT with multiple targets, the scoring function $\mathrm{R}(\hat{\mathrm{y}}_\mathrm{i},\mathrm{y_i})$ is further generalised to take several metrics into account

$$R(\hat{y}_i, y_i) = \sum_j R_j(\hat{y}_i, y_i)\beta_j \tag{6.5}$$

where β_{j} is the weight for the j-th scoring function. Since the cost function presented here (Equation 4.26) is differentiable everywhere w.r.t. model parameters θ, back propagation can be applied to calculate the gradients and update the parameters directly.

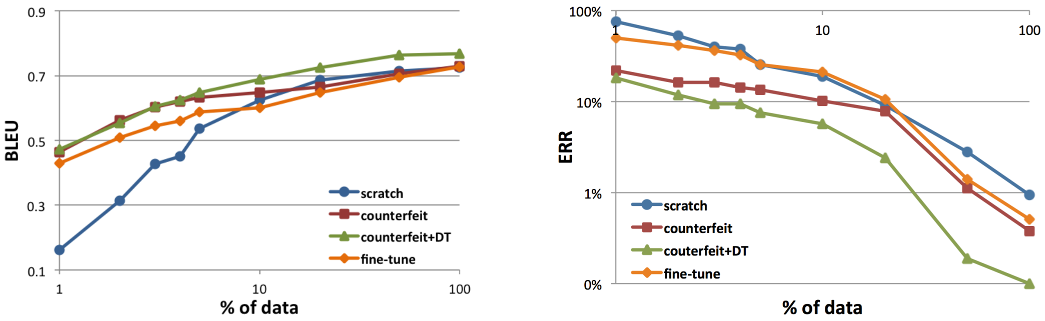
When applying the DT objective to adapt sparse target domain examples in domain adaptation scenario, a significant performance improvement was observed in almost of the cases. This is because that DT discriminates positively against the in-domain examples by not only increasing the probability of the correct examples but also penalising those incorrect ones learned in the data counterfeiting phase. This allows the model to learn biased in-domain behaviours quickly with a limited amount of in-domain data. Although this strong discriminative power learned using DT is beneficial when the amount of target domain data is limited, it is nevertheless harmful to the system trained on a vast amount of in-domain data where the discrimination is so strong that the learned system loses its ability to generate diverse responses. This is the reason why it is only used for domain adaptation rather than training NLG models in general.

6.6 Corpus-based Evaluation

6.6.1 Experimental Setup and Evaluation Metrics

To assess the effectiveness of the proposed adaptation recipe, a set of corpus-based experiments were conducted. The metrics used to evaluate the model performance are the BLEU score and slot error rate (as mentioned in Section 5.6.1). The datasets are the same as the ones described in Section 5.4, which covers the four application domains: Restaurant, Hotel, Laptop, and TV. All of these experiments use the SC-LSTM generator since it is proven to be the most robust choice across different domains (Section 5.6 and Section 5.7). The hidden layer size was set to 80 for all cases. Stochastic gradient descent, back propagation through time, and early stopping were used to optimise the parameters. For more experimental detail, such as beamwidth and decoding criteria, please refer to Section 5.6.1.

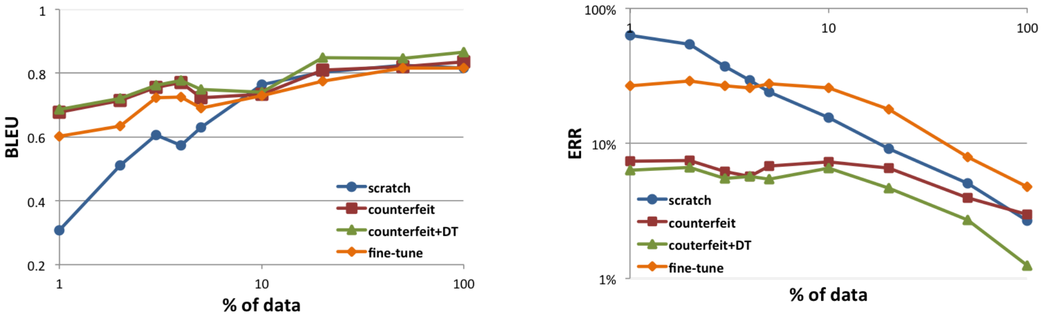
Two-way adaptation was conducted between (1) the Restaurant and Hotel domain, (2) the Laptop and TV domain, (3) the *Venue domain* formed by combining the Restaurant and Hotel domains and the *Product domain* formed by combining the Laptop and TV domains. The (1) and (2) scenarios can be considered as easier tasks than (3) because the characteristics between *venue* domains are very different from *product* domains as described in Section 5.4.3. This adaptation resulted in 6 scenarios; the results are shown in Figures 6.2 to 6.7. To facilitate comparison of the results across different adaption methods, the BLEU score and slot error rate curves are plotted for varying amounts of adaptation data. Note that the *x*-axis is presented on a log-scale in all these figures. There are two baselines: (1) the



(a) BLEU score

(b) Slot error rate

Fig. 6.2 Hotel to Restaurant domain adaptation



(a) BLEU score

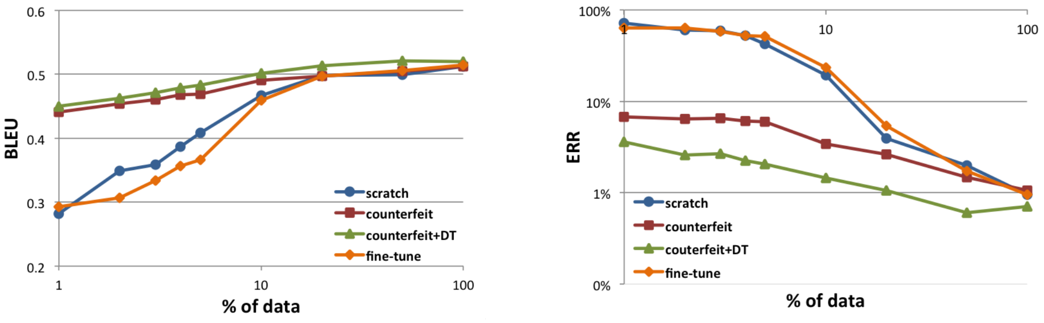
(b) Slot error rate

Fig. 6.3 Restaurant to Hotel domain adaptation

models that were trained completely from scratch using only target domain data (*scratch*), and (2) the models trained with the model fine-tuning technique (*fine-tune*) as described in Section 6.4. Therefore, the starting point of the *scratch* curve is the performance of a random initialised network, while the one for the *fine-tuned* curve indicates the performance of a model pre-trained on source domain data.

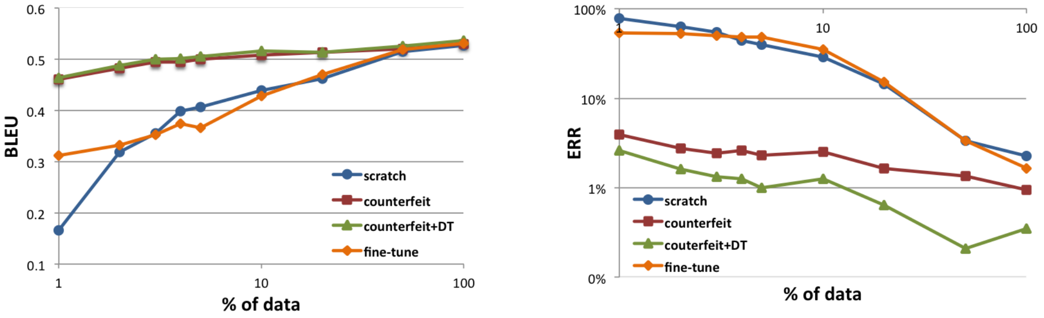
6.6.2 Results

Comparing first the data counterfeiting (*counterfeit*) approach with the two baselines (*scratch* and *fine-tune*), Figures 6.2 to 6.5 show the result of adapting models between similar domains, i.e., between Restaurant and Hotel; Laptop and TV. Because of the parameter sharing on the LM part of the network, model fine-tuning (*fine-tune*) achieves a better BLEU score than training from scratch (*scratch*), when the target domain data is limited. However, the data



(a) BLEU score (b) Slot error rate

Fig. 6.4 TV to Laptop domain adaptation

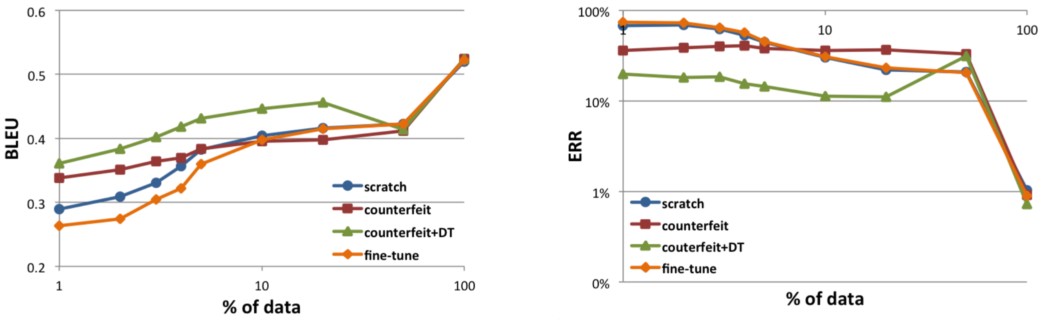


(a) BLEU score (b) Slot error rate

Fig. 6.5 Laptop to TV domain adaptation

counterfeiting (*counterfeit*) method achieves a significantly greater BLEU score than either of the baselines. This is typically because there is more robust handling of unseen slot-value pairs. Furthermore, data counterfeiting (*counterfeit*) also brings a substantial reduction in slot error rate. Counterfeiting preserves the co-occurrence statistics between slot-value pairs and realisations, which allows the model to learn good semantic alignments even before the target domain data is introduced. Similar results can be seen in Figure 6.6 and 6.7. In those figures, adaptation was performed on more disjoint domains: between Restaurant + Hotel joint domain and Laptop + TV joint domain where data counterfeiting (*counterfeit*) method is again superior to the two baselines.

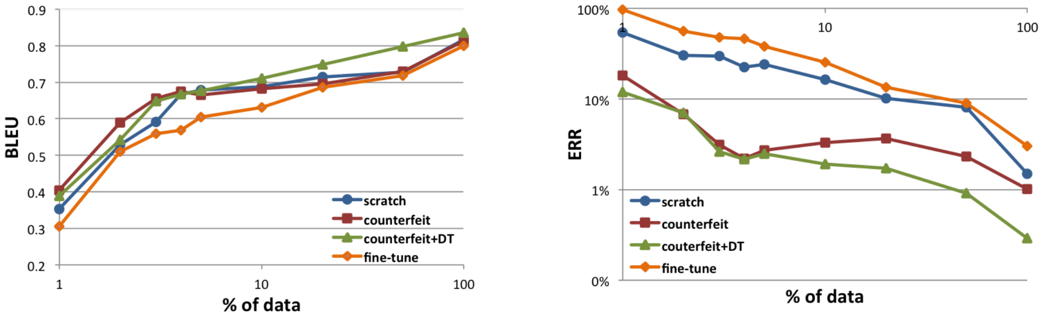
As described in Section 4.2.2, the generator parameters obtained from data counterfeiting and maximum likelihood adaptation can be further tuned by applying discriminative training. In each case, the models were optimised using two objective functions: BLEU-4 score and slot error rate. However, the sentence BLEU as introduced in Section 3.5.1 was used during training to mitigate the sparse n-gram match problem of BLEU at the sentence-level. In these experiments, we set γ to 5.0 and β_{j} to 1.0 and -1.0 (Equation 4.27 and 6.5) for BLEU and ERR, respectively. For each DA, the generator was sampled 50 times to generate candidate



(a) BLEU score

(b) Slot error rate

Fig. 6.6 Restaurant + Hotel to Laptop + TV domain adaptation



(a) BLEU score

(b) Slot error rate

Fig. 6.7 Laptop + TV to Restaurant + Hotel domain adaptation

sentences and any repeated candidates that appeared were removed. The resulting candidate set was treated as a single batch and the model parameters were updated using the procedure described in section 4.2.2.

As we can see by the curves marked *counterfeit+DT* in Figures 6.2 to 6.7, DT consistently improves generator performance for both metrics. It is interesting to note that the effect on slot error rate is more pronounced than the effect on the BLEU. This may be because the sentence BLEU optimisation criteria are only approximation to the corpus BLEU score used for evaluation. Overall, however, these results show that the proposed adaption procedure incorporates data counterfeiting and discriminative training was consistently effective across the six adaption scenarios tested.

Utterance Quality Evaluation (Rating ranges from 1 to 3)

Method	TV to Laptop		Laptop to TV	
	Informativeness	Naturalness	Informativeness	Naturalness
scr-10%	2.24^{**}	2.03^{**}	2.00^{**}	1.92^{**}
ML-10%	2.51^{**}	2.22^{**}	2.45^{**}	2.22^{**}
DT-10%	2.53^{*}	2.25^{*}	2.51	2.19^{**}
scr-All	2.64	2.37	2.54	2.36

* $p < 0.05$, ** $p < 0.005$

Table 6.1 Human evaluation of utterance quality in two adaption scenarios. Results are shown for two metrics (rated out of 3 possibilities). Statistical significance was computed using a Wilcoxon signed-rank test, between the model trained with full data (*scrALL*) and all others.

6.7 Human Evaluation

6.7.1 Experimental Setup and Evaluation Metrics

To confirm the conclusions drawn from the corpus-based evaluation, the adaptation procedure was evaluated by human judges recruited using AMT for two adaptation scenarios: laptop to TV and TV to laptop. For each task, two systems among the four were compared: training from scratch using the full dataset (*scrALL*), adapting with DT training using only 10% of the target domain data (*DT-10%*), adapting with maximum likelihood training using only 10% of the target domain data (*ML-10%*), and training from scratch using only 10% of target domain data (*scr-10%*). To reduce the effects of language variation, each system generated 5 different surface realisations for each input DA. The human judges were asked to score each of them in terms of informativeness and naturalness (rating out of 3)[3], and asked to state a preference between the two. To decrease the amount of information presented to the judges, utterances that were rendered identically in both systems were filtered out. In total, 2000 DAs were tested for each scenario and distributed uniformly between the contrast, except that 50% more comparisons were provided for the contrast between *ML-10%* and *DT-10%* because their results were very close.

6.7.2 Results

Table 6.1 shows the subjective quality assessments. We clearly see they exhibit the same general trend as in the corpus-based objective results. If a large amount of target domain

---

[3] A 3-valued likert scale was used here to reduce the cognitive load of the MTurk judges.

Pairwise Preference Comparison (%)

TV to Laptop	scr-10%	ML-10%	DT-10%	scr-All
scr-10%	-	34.5	33.9	22.4
ML-10%	65.5**	-	44.9	36.8
DT-10%	66.1**	55.1	-	35.9
scr-All	77.6**	63.2**	64.1**	-

Laptop to TV	scr-10%	ML-10%	DT-10%	scr-All
scr-10%	-	17.4	14.2	14.8
ML-10%	82.6**	-	44.9	37.1
DT-10%	85.8**	51.9	-	41.6
scr-All	85.2**	62.9**	58.4*	-

* $p < 0.05$, ** $p < 0.005$

Table 6.2 Pairwise preference test among four approaches in two adaption scenarios. Statistical significance was computed using a two-tailed binomial test.

data is available, training everything from scratch (*scrALL*) achieves very good performance and adaptation is not necessary. However, if only a limited amount of in-domain data is available, efficient adaptation is critical (*DT-10% & ML-10% > scr-10%*). Moreover, judges mostly preferred the DT adapted generator (*DT-10%*) compared to the ML adapted generator (*ML-10%*), especially for *informativeness*. In the laptop to TV scenario, the *informativeness* score for DT adaptation (*DT-10%*) was statistically indistinguishable from training on the full training set (*scrALL*).

The preference test results are shown in Table 6.2. Again, the adaptation is essential to bridge the gap between domains when the target domain data is scarce (*DT-10% & ML-10% > scr-10%*). The results also suggest that the DT training approach (*DT-10%*) was preferred compared to ML training (*ML-10%*) despite the preference being statistically insignificant. Moreover, one thing worthy of noting, the proposed adaptation recipe is quite effective. This is even the case when only 10% of the target domain data is available for adaptation. This suggests that the combination of the SC-LSTM model with the proposed adaptation recipe allows us to develop a multi-domain language generator that is more efficient by leveraging existing data at hand.

6.8 Conclusions

This chapter has presented a procedure for training multi-domain, RNN-based language generators via data counterfeiting and discriminative training. The chapter begins with a big picture description of domain adaptation. To test the efficacy of the methods, we begin by defining the three major directions: data augmentation, feature-based adaptation, and model-based adaptation.

Among the methods, it is argued that data augmentation is the most practical and efficient approach to deal with unseen slot-value pairs in the NLG problem. This leads to the data counterfeiting method, which occurs where surface forms are delexicalised. Each source domain slot-value pair is substituted with a randomly sampled target domain slot-value pair to produce pseudo examples. These pseudo examples (or counterfeited surface forms) are then used to initialise the model parameters in the target domain. Although the arbitrary replacement of slot-value pairs inevitably causes realisation errors, it does provide the necessary prior knowledge (the LM probabilities and the co-occurrence statistics of slot-value pairs and their surface forms) needed to operate the model in a new domain, even before any domain specific data is introduced.

The second phase, however, is to correct the errors introduced by the counterfeiting. By training via a more efficient objective function and leveraging the small amount of target domain data, the second phase is complete. Since the adaptation data is sparse and the maximum likelihood criterion doesn't make the most effective use of the data points to correct the model errors, the discriminative objective is instead introduced and employed.

The proposed adaptation recipe for RNNLG were assessed by both corpus-based evaluation and human evaluation. The objective measures on corpus data have demonstrated that by applying this procedure to adapting models between different dialogue domains, good performance is achieved with much less training data. Subjective assessment by human judges confirms the effectiveness of the approach. In addition, although the experiments conducted in this thesis were based on the SC-LSTM model, the procedure is general and applies to any data-driven language generator since it requires only data augmentation and a change of training objective rather than feature-level or model-level manipulation.

Chapter 7

Generation by Conditioning on a Broader Context

The framework for natural language generation described in Section 5.1 is the basis for the proposed RNNLG. The RNN model generates sentences word-by-word by conditioning on the handcrafted semantic representations such as the dialogue act formalism (Traum, 1999). The input-output space of the NLG module is designed to fit into the pipelined dialogue system framework described in Chapter 2. Thus, the system can be broken down into several components and the development can be simplified and distributed. The intermediate dialogue act representation plays a crucial role here as well; it is the major communication medium between system modules. It is also the representation that creates abstract boundaries between the various surface expressions of the same meaning and establishes the intrinsic intentions behind the conversation. Although these dialogue act taxonomies are meant to summarise the context from the front-end modules to the next system components, they were originally designed to capture just enough meaning in an utterance to provide rational system behaviours within the domain; thus, they are limited to capturing frequent yet simple conversations. Also, since there is still no agreement on what the common semantic representation should be, different annotation schemes have been applied (e.g., logical forms (Parsons, 1990) and Abstract Meaning Representation (AMR) (Chiang et al., 2013)). Although there was an effort trying to standardise the dialogue act formalism (Bunt et al., 2010), it was only used in the dialogue community due to its limited expressivity for general semantics. As a consequence, language generation by conditioning on the dialogue act representation such as the RNNLG framework in Section 5.1 is constrained by the expressive power of the handcrafted semantic representation itself and can potentially overlook many useful features in the dialogue context.

If we consider the end-to-end text-based dialogue problem as a whole, the end goal of the system is to generate a response that is appropriate given the current dialogue context. Therefore, the response generator can take the entire dialogue context as input and map it back to a natural language without the provision of the intermediate dialogue act representation. There are several hypothetical benefits for this approach:

- Firstly, it liberates the generation model from the hard-coded dialogue act representation and allows it to produce more contextually-natural replies;
- Secondly, it can model more complex dialogue scenarios and is easier to scale;
- Finally, the NLG and dialogue policy can be jointly optimised (Rieser and Lemon, 2010; Rieser et al., 2014) to avoid a suboptimal solution.

Furthermore, it also saves the additional labour of annotating the dialogue act representation which requires expert knowledge and is hard to scale even by crowdsourcing. If the representation of the dialogue context can be primitive but as complete as possible, then a neural network-based response generator should learn the appropriate dialogue intent via representations of its layers of nonlinear transformations. In this way, the semantic representation of the system can be learnt together as a latent component alongside the rest of the system module and any potentially impotent dialogue act annotations can be avoided.

7.1 End-to-End Dialogue Modelling

As mentioned in the Chapter 2, building an end-to-end goal-oriented dialogue system, such as a hotel booking or a technical support service, is difficult because it is application-specific, which results in limited training data. Therefore, machine learning approaches to goal-oriented dialogue systems typically divide the problem into modules arranged in a pipeline and cast it as a partially observable Markov Decision Process (POMDP) (Young et al., 2013). The aim of this approach is to use Reinforcement Learning (RL) to train dialogue policies online via interactions with real users (Gašić et al., 2013). Although promising results have been achieved, there are still many issues that have not been properly addressed yet:

- Firstly, although policy optimisation can be learnt quite efficiently through RL in small domains (Gašić et al., 2013, 2015), we do not know whether RL can be readily scaled to learn policies in more complex state and action settings.
- Secondly, the handcrafted nature of the state and action space (Young et al., 2013, 2010) may restrict the expressive power and learnability of the model.

- Thirdly, the reward functions needed to train such models are difficult to design and hard to measure at run-time; therefore, reward functions generally rely on explicit user feedback (Su et al., 2016b, 2015).
- Finally, even though the policy can be directly optimised online via RL, language comprehension (Henderson et al., 2014c; Yao et al., 2014) and language generation (Wen et al., 2015c, 2016b) modules still rely on supervised learning and therefore need corpora with additional semantic annotations to train on. As mentioned previously, these semantic annotations lead to additional labour with expert knowledge and cannot be easily addressed even by crowdsourcing.

At the other end of the spectrum, modelling chat-based dialogues as a sequence-to-sequence learning problem (Sutskever et al., 2014) has become a common theme in the deep learning community. This is due to the recent advances with training RNNs on a large number of corpora and the computational tractability using massive hardware. This family of approaches treats dialogues as a source that targets the sequence transduction problem. An encoder network (Cho et al., 2014) is applied to encode a user query into a distributed vector, representing its semantics, which then conditions the decoder network to generate each system response. For example, Vinyals and Le (2015) has demonstrated a seq2seq-based model trained on a huge amount of conversation corpora which learns interesting replies conditioned on different user queries, based on a qualitative assessment. It can also learn to reply to certain questions providing a reasonable answer. Similar results were also found in Shang et al. (2015) and Serban et al. (2015b). Although it is interesting to see what the model can learn by looking at just the dialogue history, the current state-of-the-art is still far from any real-world applications. First of all, these models generally suffer from the generic response problem (Li et al., 2016a; Serban et al., 2016). This is when the model struggles to produce causal or even diverse responses. Despite the fact that efforts have been made to address these issues (e.g., modelling the persona (Li et al., 2016b), reinforcement learning (Li et al., 2016c), and introducing continuous latent variables (Cao and Clark, 2017; Serban et al., 2016)), generic response remains a deep-rooted problem in many of these systems. Secondly, these models typically require a large amount of data to train but still lack any capability for supporting specific tasks. For example, specific tasks like interacting with databases (Sukhbaatar et al., 2015; Yin et al., 2015) or aggregating useful information into responses remain intractable. Lastly, the evaluation of a chat-based dialogue system is itself problematic. None of the existing automatic metrics correlate with human's perception very well (Liu et al., 2016)[1]. This is partly because of the intrinsic difficulty in evaluating NLG

---

[1] The BLEU metric they used is at sentence-level. However, experiments have shown that BLEU should be computed at the corpus-level (Papineni et al., 2002) to really reflect its effectiveness.

systems, as mentioned in Section 3.5, and partly because the chat-based scenario does not ground the conversation in any kind of knowledge or database. One thing worth to note is that when chatting to people, we humans constantly use the knowledge and experiences we have accumulated. Every conversation is more or less grounded to a knowledge, an experience, or at least the personality of the speaker. Therefore, hoping to evaluate a non-grounded chat-based NLG system via a testing corpus with a limited coverage itself doesn't make too much sense.

Another promising research direction is the Memory Network-based (MemNN) approach (Sukhbaatar et al., 2015; Weston et al., 2014) for dialogue modelling. Despite the fact that it was first proposed for Question Answering (QA), MemNN answers user queries by first embedding it as a vector. This query vector is then used to correlate with a set of supporting vectors, which are again embedded from a set of existing supporting facts, and retrieve them back via an attention mechanism. This process was extended to cater to different tasks such as factoid question answering (Miller et al., 2016), children's book reading comprehension (Hill et al., 2016), and goal-oriented dialogue (Bordes and Weston, 2017). In Bordes and Weston (2017), the dialogue problem was divided into several sub-tasks and evaluation was made on both turn-level and dialogue-level success. The major benefit of using MemNN for goal-oriented dialogue modelling, is that the entire model can be optimised end-to-end with a single objective. This approach is different from the work done in this thesis, which leverages additional supervision signals to train the explicit belief-tracking component. As a result, their MemNN model is much more data-hungry and requires more data to achieve a useful level for real-world applications. Bordes and Weston (2017) trained their model on DSTC 2 (Henderson et al., 2014a). They achieved a 40% dialogue success rate by learning from approximately 2000 distinct dialogues. The model proposed in this thesis was experimented using the same domain but only requires about 600 dialogues to achieve a success rate of 80%. Moreover, their model only learns to select a response from a set of replies. In our work, the model learns to generate words to compose sentences.

7.2 Goal-oriented Dialogue as Conditional Generation

Given the many different approaches to solve dialogue problems, basing conversations to some sort of knowledge base is a necessity for achieving rational system behaviour and producing contextual responses. For goal-oriented dialogue modelling, grounding the language on symbolic representations is important for two reasons: firstly, the symbolic representation is easier to manipulate and operate for modern computers; secondly, the symbolic representation provides a way to resolve the credit assignment problem[2] and allows the model to learn more efficiently with less data. We need to keep in mind that we also want to annotate the input via some simple scheme so that it can be readily understood by non-experts allowing crowdsourcing to be an option for data collection.

In this thesis, we propose to model a goal-oriented dialogue system[3] as a response generation problem that is conditioned on the dialogue context and a knowledge base. The dialogue context provides clues for reasonable system replies, while the knowledge base encapsulates the domain of interest that the system can talk about. Formally speaking, given the user input utterance $\mathrm{u_t}$ at turn t and the system's knowledge base (KB), the machine parses the input into a set of actionable commands Q and accesses the KB to search for useful information to reply to the query. Based on the results, the model needs to summarise its retrieved knowledge and reply with an appropriate response $\mathrm{m_t}$ in natural language. In this case, the coarse intermediate representation can just be the entities and attributes of the knowledge base. The annotations of discourse acts or the relations held between entities and attributes can simply be learnt from data. Following this framework, if we can combine a competent learning model with an efficient and simple dialogue collection procedure, we can develop a model that not only directly learns from humans or picks up interesting conversation strategies automatically, but also scales better and evolves faster.

7.3 Neural Dialogue Model

As shown in Figure 7.1, the proposed Neural Dialogue Model (NDM) treats dialogue as a sequence-to-sequence mapping problem (Sutskever et al., 2014) augmented by the dialogue history (modelled from a set of belief trackers (Henderson et al., 2014c)) and the current database search outcome (modelled from a database operator). At each turn, the system takes a sequence of tokens from the user as input and converts the tokens into two internal representations: a distributed representation generated by an intent network; and a probability distribution over slot-value pairs called the belief state (Young et al., 2013) generated by a set of belief trackers. Note, all sentences are pre-processed via delexicalisation (Henderson et al., 2014c) where slot-valued specific words are replaced with their corresponding generic tokens based on an ontology.

The database operator then selects the most probable values in the belief state to form a query for the DB. The search result, along with the representation's intent and belief state,

---

[2] The credit assignment problem concerns determining how the success of a system's overall performance is due to the various contributions of the system's components (Minsky, 1961).

[3] Like most of the goal-oriented dialogue research, we focus on information seek type dialogues.

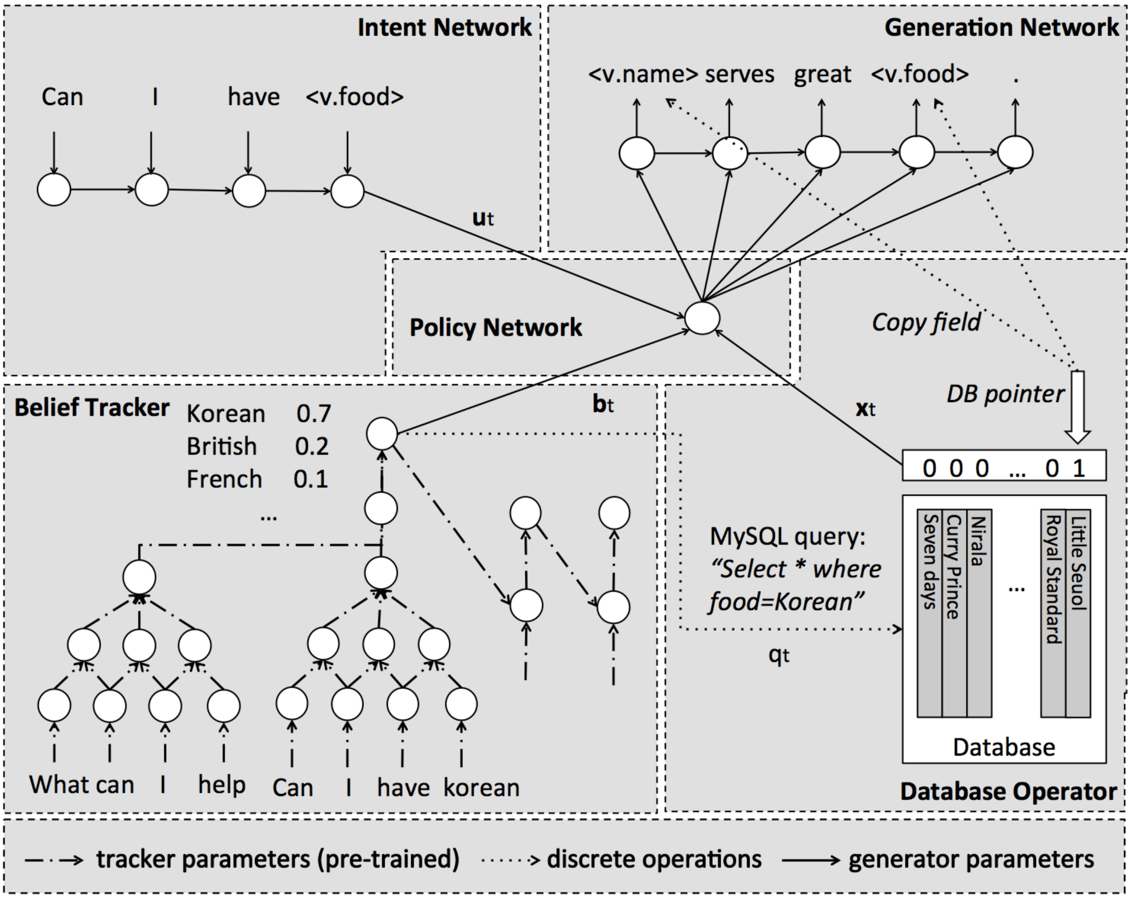


Fig. 7.1 The proposed Neural Dialogue Model framework

forms the dialogue context representation, which contains all the necessary information to decide the machine response. This context representation is then taken by an internal policy network as the input to produce a vector representing the next system's intent. This system intention vector is then used to condition a response generation network, which generates the required system output token by token in skeletal (delexicalised) form. The final system response is then formed by substituting the actual values of the database entries into the skeletal sentence structure. A more detailed description of each component is given below.

7.3.1 Intent Network

The intent network can be viewed as the encoder in the sequence-to-sequence learning framework (Sutskever et al., 2014) whose job is to encode a sequence of input tokens $\mathrm{u}_t = \{\mathrm{w}_0^t, \mathrm{w}_1^t, ...\mathrm{w}_N^t\}$ into a distributed vector representation $\mathbf{u}_t$ at every turn t. A Long Short-term Memory (LSTM) network (Hochreiter and Schmidhuber, 1997) is typically used and

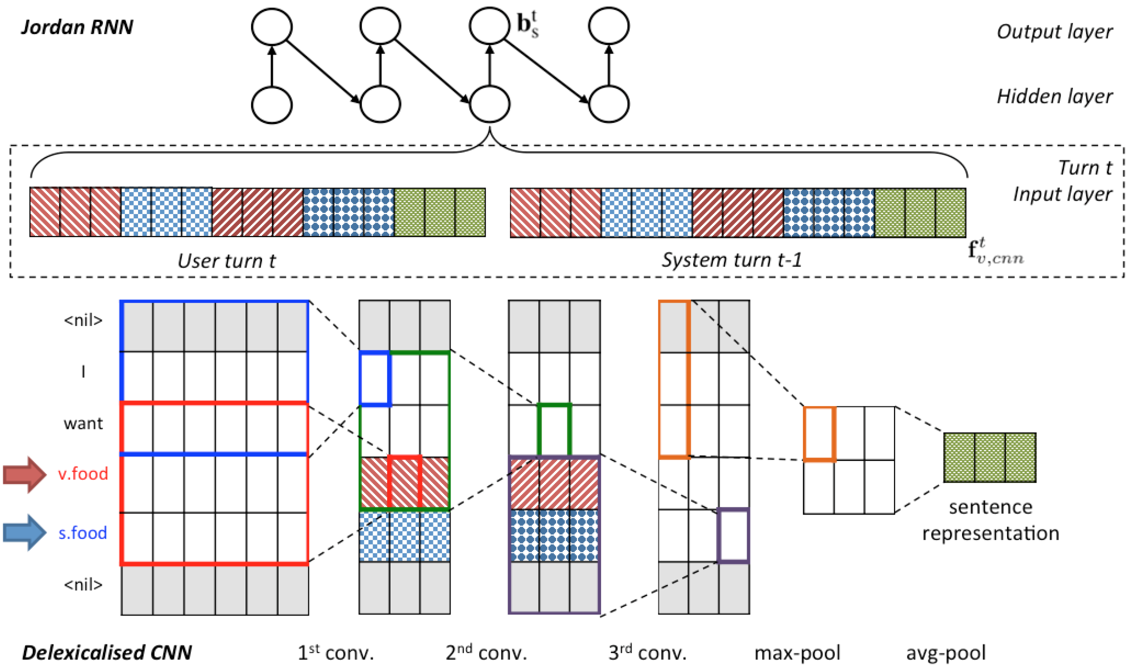


Fig. 7.2 Tied Jordan-type RNN belief tracker with delexicalised CNN feature extractor. The output of the CNN feature extractor is a concatenation of top-level sentence (green) embedding and several levels of intermediate n-gram-like embedding's (red and blue). However, if a value cannot be delexicalised in the input, its ngram-like embedding's will be padded with zeros. We zero-pad vectors (in gray) before each convolution operation to make sure the representation at each layer has the same length. The output of each tracker $\mathbf{b}^t_s$ is a distribution over values of a particular slot s.

the previous time step's hidden layer $\overrightarrow{\mathbf{u}_t}[\mathrm{N}]$ is taken as the representation

$$\mathbf{u}_t = \overrightarrow{\mathbf{u}_t}[N] = \overrightarrow{\mathrm{LSTM}}(u_t)[N]. \tag{7.1}$$

However, bidirectional LSTM, which uses two LSTMs to read the input sentence from both directions, is often used because it offers a more integral view of the input and mitigates the vanishing gradient problem automatically

$$\mathbf{u}_t = \mathrm{biLSTM}(u_t) = \overrightarrow{\mathrm{LSTM}}(u_t)[N] \oplus \overleftarrow{\mathrm{LSTM}}(u_t)[N]. \tag{7.2}$$

Therefore, throughout this thesis, all the intent networks are encoded using bidirectional LSTMs. Since slot-value specific information is delexicalised, the encoded vector can be viewed as a distributed intent representation, which replaces the hand-coded dialogues act representation (Traum, 1999) from traditional goal-oriented dialogue systems.

7.3.2 The RNN-CNN Dialogue State Tracker

Belief tracking (also called Dialogue State tracking), as discussed in Section 2.3, provides the foundation of a goal-oriented dialogue system (Henderson, 2015b). Current state-of-the-art belief trackers use discriminative models, such as Recurrent Neural Networks, to directly map ASR hypotheses to belief states (Henderson et al., 2014c; Mrkšić et al., 2016b). Although we focus on text-based dialogue systems in this work, we retain belief tracking at the core of our system because firstly, belief tracking enables a sequence of free-form natural language sentences that are mapped into a fixed set of slot-value pairs, which can then be used to query a DB. This mapping can be viewed as a simple version of a semantic parser (Berant et al., 2013); secondly, by keeping track of the dialogue state, belief tracking avoids learning unnecessarily complicated long-term dependencies from raw inputs; thirdly, belief tracking uses a smart weight-tying strategy that can greatly reduce the data needed to train the model, and lastly, belief tracking provides an inherent robustness, which simplifies future extensions to spoken systems.

Taking each user input as new evidence, the task of a belief tracker is to maintain a multinomial distribution b over values $\mathrm{v} \in \mathrm{V_s}$ for each informable slot s and a binary distribution for each requestable slot. Note, Informable slots are slots that users can use to constrain the search, such as food type or price range, while Requestable slots are slots that users can ask a value for, such as address. Each slot in the ontology $\mathbb{G}$[4] has its own specialised tracker. Each tracker is a Jordan-type (recurrence from output to hidden layer) (Jordan, 1989) RNN[5] tracker with a CNN-based feature extractor, as shown in Figure 7.2. As in Mrkšić et al. (2015), we tie the RNN weights together for each value of v but vary the $\mathbf{f}_\mathrm{v}^\mathrm{t}$ features when updating each pre-softmax activation $\mathrm{g}_\mathrm{v}^\mathrm{t}$. The update equations for a given slot s are,

$$\mathbf{f}_\mathrm{v}^\mathrm{t} = \mathbf{f}_\mathrm{v,cnn}^\mathrm{t} \oplus \mathrm{p}_\mathrm{v}^{\mathrm{t}-1} \oplus \mathrm{p}_\emptyset^{\mathrm{t}-1} \tag{7.3}$$

$$\mathrm{g}_\mathrm{v}^\mathrm{t} = \mathbf{w}_\mathrm{s} \cdot \mathrm{sigmoid}(\mathbf{W}_\mathrm{s}\mathbf{f}_\mathrm{v}^\mathrm{t} + \mathbf{l}_\mathrm{s}) + \mathrm{l}'_\mathrm{s} \tag{7.4}$$

$$\mathrm{b}_\mathrm{v}^\mathrm{t} = \frac{\exp(\mathrm{g}_\mathrm{v}^\mathrm{t})}{\exp(\mathrm{g}_{\emptyset,\mathrm{s}}) + \sum_{\mathrm{v}' \in \mathrm{V_s}} \exp(\mathrm{g}_{\mathrm{v}'}^\mathrm{t})}, \tag{7.5}$$

where the vector is characterised by $\mathbf{w}_\mathrm{s}$, the matrix is $\mathbf{W}_\mathrm{s}$, bias terms is $\mathbf{l}_\mathrm{s}$ and l'_s, and the scalar $\mathrm{g}_{\emptyset,\mathrm{s}}$ are parameters. $\mathrm{b}_\emptyset^\mathrm{t}$ is the probability that the user has not mentioned that slot, up to turn t, and can be calculated by substituting $\mathrm{g}_{\emptyset,\mathrm{s}}$ for $\mathrm{g}_\mathrm{v}^\mathrm{t}$ in the numerator of Equation 7.5. Note that the top-level weight vector $\mathbf{w}_\mathrm{s}$ is tied to each individual value v; therefore, it can learn to identify rare or even unseen values by leveraging training data points of the other values.

---

[4] A small knowledge graph defining the slot-value pairs that the system can talk about, for a particular task.

[5] We don't use the recurrent connection for requestable slots since they do not need to be tracked.

To model the discourse context at each turn, the feature vector $\mathbf{f}^{t}_{\mathrm{v,cnn}}$ is the concatenation of two CNN derived features, one from processing the user input $\mathrm{u_t}$ at turn t and the other from processing the machine response $\mathrm{m_{t-1}}$ at turn $\mathrm{t}-1$,

$$\mathbf{f}^{t}_{v,cnn} = \mathrm{CNN}^{(u)}_{s,v}(u_t) \oplus \mathrm{CNN}^{(m)}_{s,v}(m_{t-1}), \tag{7.6}$$

where every token in $\mathrm{u_t}$ and $\mathrm{m_{t-1}}$ is represented by an embedding of size N, derived from a 1-hot input vector. This can be visualised in the tracker block of Figure 7.1, where two CNNs are used to process the sentences. To make the tracker aware of delexicalisation after it is applied to a slot or value, the slot-value specialised CNN operator $\mathrm{CNN}^{(\cdot)}_{\mathrm{s,v}}(\cdot)$ extracts not only the top level sentence representation, but also an intermediate n-gram-like embedding determined by the position of the delexicalised token in each utterance. If multiple matches are observed, each corresponding embedding is summed. On the other hand, if there is no match for the slot or value, each empty n-gram embedding is padded with zeros. To keep track of the position of delexicalised tokens, both sides of the sentence are padded with zeros before each convolution operation to make each intermediate representation the same size as the original input. The number of vectors is determined by the filter size at each layer. The overall process of extracting several layers of position-specific features is visualised in Figure 7.2.

The belief tracker described above is based on Henderson et al. (2014c), with some modifications: firstly, only probabilities over informable and requestable slots and values are output. This is because we want to keep the annotations as simple as possible; secondly, the recurrent memory block is removed, since it appears to offer no benefit in this task; and thirdly, the n-gram feature extractor is replaced by the CNN extractor described above to allow for more flexible feature extractions. By introducing slot-based belief trackers, we essentially add a set of intermediate labels into the system, compared to training a pure end-to-end system. These tracker components have been shown to be critical for achieving task success (Wen et al., 2017b). We will also show that the additional annotation effort they introduce can be successfully mitigated using a novel pipelined Wizard-of-Oz (WoZ) data collection framework.

7.3.3 Database Operator and Deterministic Policy Network

Database Operator

Based on the output $\mathbf{b}_\mathrm{s}^\mathrm{t}$ of the belief trackers, the DB query q_t is formed by

$$q_t = \bigcup_{s' \in S_I} \{\underset{v}{\operatorname{argmax}}\, \mathbf{b}_{s'}^{t}\}, \tag{7.7}$$

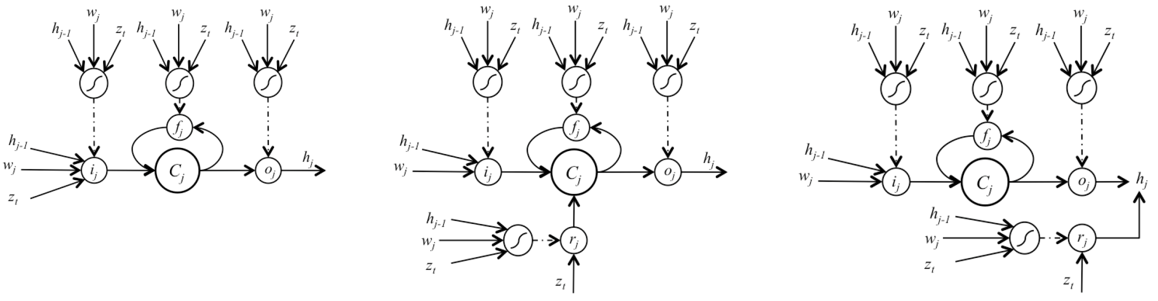
where S_I is the set of informable slots. This query is then applied to the DB to create a binary truth-value vector $\mathbf{x}_\mathrm{t}$ over DB entities where a 1 indicates that the corresponding entity is consistent with the query (and hence it is consistent with the most likely belief state). In addition, if $\mathbf{x}$ is not entirely null, an associated entity pointer is maintained. This identifies one of the matching entities selected at random. The entity pointer is updated if the current entity no longer matches the search criteria; otherwise it remains the same. The entity referenced by the entity pointer is used to form the final system response as described in Section 7.3.4.

Policy Network

The policy network can be viewed as the decision-making centre that takes the dialogue context and transforms it into an action representation of the system. Its output is a single vector $\mathbf{z}_\mathrm{t}$ representing the system action. Its inputs are comprised of $\mathbf{u}_\mathrm{t}$ from the intent network, the belief state $\mathbf{b}_\mathrm{s}^\mathrm{t}$, and the DB truth value vector $\mathbf{x}_\mathrm{t}$. Since the generation network only generates appropriate sentence forms, the individual probabilities of the categorical values in the informable belief state are immaterial. They are summed together to form a summary belief vector for each slot $\hat{\mathbf{b}}_\mathrm{s}^\mathrm{t}$, which is represented by three components: the summed value probabilities, the probability that the user said they "don't care" about this slot and the probability that the slot has not been mentioned. Similarly for the truth value vector $\mathbf{x}_\mathrm{t}$, the number of matching entities matters but not their identity. This vector is therefore compressed to a 6-bin 1-hot encoding $\hat{\mathbf{x}}_\mathrm{t}$, which represents varying degrees of matching in the DB (no match, 1 match, ... or more than 5 matches). Finally, the policy network output is generated by a three-way matrix transformation

$$\mathbf{z}_t = \tanh(\mathbf{W}_{uz}\mathbf{u}_t + \mathbf{W}_{bz}\hat{\mathbf{b}}_t + \mathbf{W}_{xz}\hat{\mathbf{x}}_t), \tag{7.8}$$

where matrices $\{\mathbf{W}_\mathrm{uz}, \mathbf{W}_\mathrm{bz}, \mathbf{W}_\mathrm{xz}\}$ are parameters and $\hat{\mathbf{b}}_\mathrm{t} = \bigoplus_{\mathrm{s}\in\mathbb{G}} \hat{\mathbf{b}}_\mathrm{s}^\mathrm{t}$ is a concatenation of all summary belief vectors. This can be viewed as deterministic policy modelling because the policy is completely based on the input.



(a) Language model type LSTM (b) Memory type LSTM (c) Hybrid type LSTM

Fig. 7.3 Three different conditional generation architectures.

Attentive Policy Network

An attention-based mechanism provides an effective approach for aggregating multiple information sources for prediction tasks. Instead of just using a simple deterministic matrix transformation for decision-making, we explore the use of an attention mechanism to combine the tracker belief states in which the policy network in Equation 7.8 is modified as

$$\mathbf{z}_{\mathrm{t}}^{\mathrm{j}} = \tanh(\mathbf{W}_{\mathrm{uz}}\mathbf{u}_{\mathrm{t}} + \mathbf{W}_{\mathrm{xz}}\mathbf{x}_{\mathrm{t}} + \textstyle\sum_{\mathrm{s}\in\mathbb{G}} \alpha_{\mathrm{s}}^{\mathrm{j}}\mathbf{W}_{\mathrm{bz}}^{\mathrm{s}}\mathbf{b}_{\mathrm{t}}^{\mathrm{s}}),$$

where j is the generation step of the decoder and the attention weights $\alpha_{\mathrm{s}}^{\mathrm{j}}$ are calculated via

$$\alpha_{\mathrm{s}}^{\mathrm{j}} = \mathrm{softmax}\big(\mathbf{r}^{\mathsf{T}}\tanh\big(\mathbf{W}_{\mathrm{r}}\cdot(\mathbf{v}_{\mathrm{t}} \oplus \mathbf{b}_{\mathrm{t}}^{\mathrm{s}} \oplus \mathbf{w}_{\mathrm{j}}^{\mathrm{t}} \oplus \mathbf{h}_{\mathrm{j-1}}^{\mathrm{t}})\big)\big),$$

where $\mathbf{v}_{\mathrm{t}} = \mathbf{u}_{\mathrm{t}} + \mathbf{x}_{\mathrm{t}}$, $\mathbf{w}_{\mathrm{j}}^{\mathrm{t}}$ and $\mathbf{h}_{\mathrm{j-1}}^{\mathrm{t}}$ are current-step word-embeddings and previous step decoder context, respectively, and $\{\mathbf{W}_{\mathrm{r}}, \mathbf{r}\}$ are learning parameters. By adopting an attention mechanism in the policy network, the action embedding is no longer constant during decoding. This allows the generator to selectively focus on part of the input tracker space rather than conditioning over the entire belief state while generating the next word. The action embedding $\mathbf{z}_{\mathrm{t}}^{\mathrm{j}}$ remains solely determined by the input, and therefore the decision variations in the data or any intrinsic agent decisions cannot be modelled yet.

7.3.4 Conditional LSTM Generator and Its Variants

Conditioned on the system action representation $\mathbf{z}_{\mathrm{t}}$, which is provided by the policy network, the decoder module uses a conditional LSTM LM to generate the required system output token by token in skeletal form[6]. The final system response can then be formed by substituting

---

[6] The output is delexicalised by replacing slot-value specific realisations with generic tokens.

the actual values of the database entries into the skeletal sentence structure. This is different from the situation we faced in Section 5.1 where the dialogue act taxonomies serve as an intermediate representation. These taxonomies can be easily broken down into several meaningful pieces (slot and values) and tackled individually using separate sigmoid gates. This is done while the dialogue act representation is in a high-dimensional, continuous space, which is hard to interpret. Therefore, we now give up the SC-LSTM-like explicit gating controls but instead simply condition the LSTM decoder on the dialogue act vector. In this thesis, we study and analyse three different variants of LSTM-based conditional generation architectures: the LM type, the memory type, and the hybrid type decoder. More details are provided in the sections that follow.

Language Model Type Decoder

The most straightforward way to condition the LSTM network on additional source information is to concatenate the conditioning vector $\mathbf{z}_t$ together with the input word embedding $\mathbf{w}_j$ and the previous hidden layer $\mathbf{h}_{j-1}$,

$$\begin{pmatrix} \mathbf{i}_j \\ \mathbf{f}_j \\ \mathbf{o}_j \\ \hat{\mathbf{c}}_j \end{pmatrix} = \begin{pmatrix} \text{sigmoid} \\ \text{sigmoid} \\ \text{sigmoid} \\ \tanh \end{pmatrix} \mathbf{W}_{4n,3n} \begin{pmatrix} \mathbf{z}_t \\ \mathbf{w}_j \\ \mathbf{h}_{j-1} \end{pmatrix}$$

$$\mathbf{c}_j = \mathbf{f}_j \odot \mathbf{c}_{j-1} + \mathbf{i}_j \odot \hat{\mathbf{c}}_j$$

$$\mathbf{h}_j = \mathbf{o}_j \odot \tanh(\mathbf{c}_j),$$

where the index j is the generation step, n is the hidden layer size, $\mathbf{i}_j, \mathbf{f}_j, \mathbf{o}_j \in [0,1]^n$ are input, forget, and output gates, respectively, $\hat{\mathbf{c}}_j$ and $\mathbf{c}_j$ are proposed cell values and true cell values at step j. While $\mathbf{W}_{4n,3n}$ are model parameters. The model is shown in Figure 7.3a. Since the resulting model does not differ significantly from the original LSTM, we call it the *language model type* (lm) conditional generation network.

Memory Type Decoder

The *memory type* (mem) conditional generation network was introduced by Wen et al. (2015c). The network is shown in Figure 7.3b. We see that the conditioning vector $\mathbf{z}_t$ is governed by a standalone reading gate $\mathbf{r}_j$. This reading gate decides how much information should be read from the conditioning vector and directly writes it into the memory cell $\mathbf{c}_\mathrm{j}$,

$$\begin{pmatrix} \mathbf{i}_\mathrm{j} \\ \mathbf{f}_\mathrm{j} \\ \mathbf{o}_\mathrm{j} \\ \mathbf{r}_\mathrm{j} \end{pmatrix} = \begin{pmatrix} \mathrm{sigmoid} \\ \mathrm{sigmoid} \\ \mathrm{sigmoid} \\ \mathrm{sigmoid} \end{pmatrix} \mathbf{W}_{\mathrm{4n,3n}} \begin{pmatrix} \mathbf{z}_\mathrm{t} \\ \mathbf{w}_\mathrm{j} \\ \mathbf{h}_{\mathrm{j}-1} \end{pmatrix}$$

$$\hat{\mathbf{c}}_\mathrm{j} = \tanh\big(\mathbf{W}_\mathrm{c}(\mathbf{w}_\mathrm{j} \oplus \mathbf{h}_{\mathrm{j}-1})\big)$$

$$\mathbf{c}_\mathrm{j} = \mathbf{f}_\mathrm{j} \odot \mathbf{c}_{\mathrm{j}-1} + \mathbf{i}_\mathrm{j} \odot \hat{\mathbf{c}}_\mathrm{j} + \mathbf{r}_\mathrm{j} \odot \mathbf{z}_\mathrm{t}$$

$$\mathbf{h}_\mathrm{j} = \mathbf{o}_\mathrm{j} \odot \tanh(\mathbf{c}_\mathrm{j}),$$

where $\mathbf{W}_\mathrm{c}$ is another weight matrix the network needs to learn. The idea behind this is that the model isolates the conditioning vector from the LM so that the model has more flexibility to learn to trade-off between the two.

Hybrid Type Decoder

Continuing with the same idea as the *memory type* network, a complete separation of conditioning vector and LM (except for the gate controlling the signals) is provided by the *hybrid-type* network shown in Figure 7.3c,

$$\begin{pmatrix} \mathbf{i}_\mathrm{j} \\ \mathbf{f}_\mathrm{j} \\ \mathbf{o}_\mathrm{j} \\ \mathbf{r}_\mathrm{j} \end{pmatrix} = \begin{pmatrix} \mathrm{sigmoid} \\ \mathrm{sigmoid} \\ \mathrm{sigmoid} \\ \mathrm{sigmoid} \end{pmatrix} \mathbf{W}_{\mathrm{4n,3n}} \begin{pmatrix} \mathbf{z}_\mathrm{t} \\ \mathbf{w}_\mathrm{j} \\ \mathbf{h}_{\mathrm{j}-1} \end{pmatrix}$$

$$\hat{\mathbf{c}}_\mathrm{j} = \tanh\big(\mathbf{W}_\mathrm{c}(\mathbf{w}_\mathrm{j} \oplus \mathbf{h}_{\mathrm{j}-1})\big)$$

$$\mathbf{c}_\mathrm{j} = \mathbf{f}_\mathrm{j} \odot \mathbf{c}_{\mathrm{j}-1} + \mathbf{i}_\mathrm{j} \odot \hat{\mathbf{c}}_\mathrm{j}$$

$$\mathbf{h}_\mathrm{j} = \mathbf{o}_\mathrm{j} \odot \tanh(\mathbf{c}_\mathrm{j}) + \mathbf{r}_\mathrm{j} \odot \mathbf{z}_\mathrm{t}$$

This model was motivated by the fact that long-term dependency is not needed by the conditioning vector because we apply this information at every step j anyway. The decoupling of the conditioning vector and LM is attractive because it leads to better interpretability of the results and provides the potential to learn from isolated conditioning vectors and LMs.

7.4 Wizard-of-Oz Data Collection

Arguably the greatest bottleneck for statistical approaches to dialogue system development is the collection of appropriate training data. This is especially true for task-oriented dialogue systems. Serban et al. (2015a) have catalogued existing corpora to develop conversational agents. Such corpora may also be useful for bootstrapping. For task-oriented dialogue systems, in-domain data is essential[7]. To mitigate this problem, we propose a novel crowdsourcing version of the Wizard-of-Oz (WoZ) paradigm (Kelley, 1984) for collecting domain-specific corpora.

Based on the given ontology, we designed two webpages on the Amazon Mechanical Turk, one for wizards and the other for users (see Appendix D for the designs). The users are given a task that specifyies the characteristics of a particular entity that they must find (e.g. *a Chinese restaurant in the north*). They are then asked to type in natural language sentences to fulfil the task. The wizards are given a form to record the information conveyed in the last user's turn (e.g. *pricerange=Chinese, area=north*). They are also provided with a search table showing all the available matching entities in the database. We note that these forms contain all the labels needed to train the slot-based belief trackers. The table is automatically updated every time the wizard submits new information. Based on the updated table, the wizard types an appropriate system response and the dialogue continues.

To enable large-scale, parallel data collection and avoid the distracting latencies inherent in conventional WoZ scenarios (Bohus and Rudnicky, 1999), users and wizards are asked to contribute only a single turn to each dialogue. To ensure coherence and consistency, users and wizards must review all previous turns in the dialogue before they contribute their turns. Thus, dialogues progress in a pipeline. Many dialogues can be active in parallel. No worker needs to wait for a response from the other party in the dialogue. Even though multiple workers contribute to each dialogue, we observe that dialogues are generally coherent, but diverse. To ensure the quality of the collected data, two things are especially important:

- Setting a high bar for worker qualifications[8].
- Collecting data iteratively and rejecting HITs if the collected example is unacceptable.

Furthermore, the turn-level data collection strategy seems to encourage workers to learn and correct each other based on previous turns.

In this thesis, the system was designed to assist users to find a restaurant in the Cambridge, UK area. There are three informable slots (*food*, *pricerange*, *area*), which users can use to

---

[7] e.g., technical support for Apple computers may differ completely from that for Windows, due to the many differences in software and hardware.

[8] In our case, setting the approval rate above 97% works the best.

constrain the search. There are six requestable slots (*address*, *phone*, *postcode* plus the three informable slots) that the user can ask about once a restaurant has been offered. There are 99 restaurants in the DB. Based on this domain, we ran 3000 HITs (Human Intelligence Tasks) in total for roughly 3 days and collected roughly 2750 dialogue turns (counting both user and system turns). After cleaning the data, we have approximately 680 dialogues in total (some of them are unfinished). The total cost for collecting this dataset was $\sim$ 400 USD.

7.5 Evaluation on Neural Dialogue Models

To evaluate the effectiveness of the proposed NDM as well as the WoZ data collection, a corpus-based evaluation and a human evaluation were performed. Detailed are provided in the next section.

7.5.1 Experimental Setup and Evaluation Metrics

Training Training of the NDM is divided into two phases. Firstly the belief tracker parameters θ_b are trained using the cross entropy errors between tracker labels $\mathbf{y}_\mathrm{s}^\mathrm{t}$ and predictions $\mathbf{b}_\mathrm{s}^\mathrm{t}$, $\mathrm{L}_1(\theta_\mathrm{b}) = -\sum_\mathrm{t}\sum_\mathrm{s}(\mathbf{y}_\mathrm{s}^\mathrm{t})^\intercal \log \mathbf{b}_\mathrm{s}^\mathrm{t}$. For the full model, we have three informable trackers (*food, price range, and area*) and six requestable trackers (*address, phone, postcode*, plus the three informable slots). An additional tracker was also trained to detect user goal changes (handling queries like "*is there anything else?*").

Having fixed the tracker parameters, the remaining parts of the model $\theta_{\backslash\mathrm{b}}$ are trained using the cross entropy errors from the generation network language model, $\mathrm{L}_2(\theta_{\backslash\mathrm{b}}) = -\sum_\mathrm{t}\sum_\mathrm{j}(\mathbf{y}_\mathrm{j}^\mathrm{t})^\intercal \log \mathbf{p}_\mathrm{j}^\mathrm{t}$, where $\mathbf{y}_\mathrm{j}^\mathrm{t}$ and $\mathbf{p}_\mathrm{j}^\mathrm{t}$ are output as token targets and predictions, respectively, at turn t of output step j. We treated each dialogue as a batch and used the Adam optimiser (Kingma and Ba, 2014) with a small l2-regularisation term to train the model. The collected corpus was partitioned into training, validation, and testing sets in the ratio 3:1:1. Early stopping was implemented and based on the validation set for which the maximum degradation multiplier was set to 2 and gradient clipping was set to 1. All the hidden layer sizes were set to 50 and all the weights were randomly initialised between -0.3 and 0.3. This included the initialisation of each word embedding. While initialising both the LSTM encoder and decoder, we initialise the bias term of the LSTM forget gate and set its value to 2, as suggested in Jozefowicz et al. (2015). The vocabulary size is approximately 500 for both input and output. Rare words and words that can be delexicalised are removed. We used three convolutional layers for all belief tracking CNNs and all the filter sizes were set to 3. Pooling operations were only applied after the final convolution layer.

Decoding To decode without length bias, we decoded each system's response $\mathrm{m_t}$ based on the average log probability of tokens

$$m_t^* = \underset{m_t}{\operatorname{argmax}}\{\log p(m_t|\theta, u_t)/\|m_t\|\}, \tag{7.9}$$

where θ is the model parameter. As studied in Wen et al. (2017b), the repetition of slot or value tokens can hurt the model because it may generate placeholders that cannot be properly realised. Therefore, we introduce an additional penalty function for repeated tokens that carry slot-value specific information and modify the decoding criterion

$$m_t^* = \underset{m_t}{\operatorname{argmax}}\{\log p(m_t|\theta, u_t)/\|m_t\| + \alpha R_t\}, \tag{7.10}$$

where α is the trade-off parameter and was empirically set to 1.0. We used a simple heuristic for the scoring function $\mathrm{R_t}$ as

$$R_t = \begin{cases} -\infty & \text{if a value token repeats,} \\ -\frac{1}{2}(\mathrm{r}-1)^2 & \text{if a slot token repeats,} \end{cases} \tag{7.11}$$

where r is the number of repetitions of that token. Note Equation 7.11 allows slot token repetitions, to some degree, when the loss grows in a polynomial way. However, the heuristic cannot tolerate value token repetitions, which are basically cut down from the branch during decoding simply because the model won't be able to properly realise these additional value-specific tokens. Another topic worth mentioning is that, unlike Wen et al. (2017b), we do not apply a fully-weighted decoding strategy here because we want to keep the decoding as simple as possible so that the performance gain is mainly from the intrinsic capabilities of the model. Furthermore, the MMI criterion (Li et al., 2016a) was proven to be ineffective for this task. During decoding, the beamwidth was set to 10 and the search stops when an *end of sentence token* is generated.

Metrics We compared models trained with different recipes by performing a corpus-based evaluation. The model is used to predict each system response in the held-out test set. Three evaluation metrics were used: BLEU score on the top-1 generated candidates (Papineni et al., 2002) and the objective task success rate (Su et al., 2015). The dialogue is marked as successful if the following two conditions are true: (1) the offered entity matches the task that was specified by the user, and (2) the system answered all the associated information requests (e.g., *what is the address?*) from the user. We computed the BLEU scores of the skeletal sentence forms before substituting the actual entity values. A weighted metric of BLEU, and success rate, was also computed by multiplying the BLEU score with 0.5 and

Table 7.1 Corpus-based experiment by comparing different NDM architectures. The results were obtained by training models with several hyper-parameter settings. The testing performance is based on the best model of the validation set.

Model	Decoder	Success(%)	BLEU	Suc.+0.5 BLEU
Vanilla NDM	lm	72.8%	0.237	0.847
	mem	74.3%	0.243	0.865
	hybrid	77.9%	0.231	0.894
Attentive NDM	lm	72.1%	0.246	0.844
	mem	80.1%	0.240	0.921
	hybrid	77.9%	0.234	0.896

Table 7.2 Human assessment of NDM. The rating for comprehension and naturalness are both out of 5.

Metric	NDM
Success	98%
Comprehension	4.11
Naturalness	4.05
# of dialogues:	245

Table 7.3 A comparison of the NDM with a rule-based modular system (*HDC*). A two-tailed binomial test was used.

Metric	NDM	HDC	Tie
Subj. Success	96.95%	95.12%	-
Avg. # of Turn	3.95	4.54	-
Comparisons(%)			
Naturalness	46.95^*	25.61	27.44
Comprehension	45.12^*	21.95	32.93
Preference	50.00^*	24.39	25.61
Performance	43.90^*	25.61	30.49

* p <0.005, # of comparisons: 164

summing the two metrics together. This serves as the main indicator when comparing models because we care about the success rate slightly more than the BLEU rate, although both are important.

7.5.2 Results

The corpus-based evaluation results are shown in Table 7.1. Since neural networks are usually sensitive to random seeds and hyper-parameters, we ran a small grid search over the following hyper-parameters: the initial learning rate of Adam, with range $[0.008, 0.010, 0.012]$; the l2 regularisation terms, with range $[0.0, 1e-6, 1e-5, 1e-4]$; and the random seed with values ranging from 1 to 5. We trained a model based on each of the hyper-parameter combinations mentioned above. We picked the model which performed the best on the validation set, and reported its performance on the testing set.

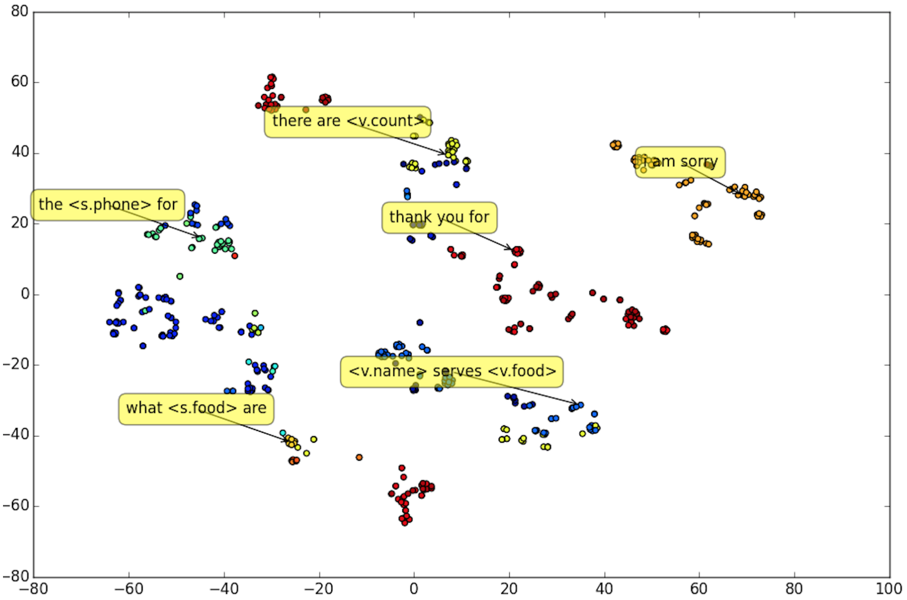


Fig. 7.4 The action vector embedding $\mathbf{z}_t$ generated by the vanilla NDM model. Each cluster is labelled with the first three words that the embedding generated.

As suggested in Table 7.1, the attentive NDMs generally perform better than the vanilla NDMs, in terms of both metrics. This is mainly due to the inclusion of an attention mechanism, which gives the model the freedom to change focus while generating the sentence. This is consistent with the findings of Wen et al. (2016a). However, there is no clear "best choice" for the language decoder, even though Wen et al. (2016a) suggested that the *hybrid* type LSTM decoder, which uses a complete separation of conditional vector and language model, yielded the best result. This discrepancy can be attributed to two major reasons: firstly, in Wen et al. (2016a) the encoder used was a single-direction LSTM. In this thesis, however, a bidirectional LSTM encoder is used; secondly, all the LSTM-based forget gate biases were set to 2 as suggested in Jozefowicz et al. (2015), which were not employed by Wen et al. (2016a). Because both of the methods mentioned above were trying to address the same vanishing gradient problem (Bengio et al., 1994) when training RNN, the effectiveness of the decoder separation is therefore offset.

7.5.3 Human Evaluation

To assess operational performance, we tested our model using paid subjects recruited via Amazon Mechanical Turk. Each judge was asked to perform a task and to rate the model's performance. We assessed the subjective success rate, the perceived comprehension ability, and naturalness of response on a scale of 1 to 5. The attentive NDM was used and the system was tested on a total of 245 dialogues. As we see in Table 7.2, the average subjective success rate was 98%. This result suggests that the system was able to complete the majority of tasks. Moreover, the comprehension ability and naturalness scores both averaged more than 4 out of 5. (See Table 7.4 for sample dialogues in this trial.)

We also ran comparisons between the NDM and a handcrafted, modular baseline system (*HDC*); it consists of a handcrafted semantic parser, rule-based policy and belief tracker, and a template-based generator. The results can be seen in Table 7.3. The HDC system achieved a $\sim 95\%$ task success rate, which suggests that it is a strong baseline, even though most of the components were hand-engineered. Over the 164-dialogues tested, the NDM was considered better than the handcrafted system on all the metrics compared. Although both systems achieved similar success rates, the NDM was more efficient and provided a more engaging conversation (lower turn number and higher preference). Moreover, the comprehension ability and naturalness of the NDM were also rated much preferred, which suggests that the learned system was perceived as being more natural than the hand-designed system.

Action Embedding A major argument for learning end-to-end dialogue systems is that we can directly learn from human-human conversations without the need for explicit handcrafting of the semantic representation, such as dialogue acts. Consequently, analysing what the NDM learns as its internal semantic representation is an important task still left to do. To better understand the meaning of the conditional vector $\mathbf{z}_t$ represents, we used t-SNE (der Maaten and Hinton, 2008) to produce a reduced dimensional view of the $\mathbf{z}_t$ embedding, plotted and labelled by the first three generated output words[9]. The result is shown as Figure 7.4. We can see soft intent clusters scattering in different places, even though we did not explicitly model them using DAs. This shows the main advantage of NDM: it learns to group states and actions that are supposed to close to each other together via direct supervision signal from the output responses, which helps the dialogue policy generalises through the learnt high dimensional state and action space.

7.6 Conclusions

In this chapter, we estimate the possibility of training a dialogue response generator via conditioning on a broader context without the provision of a handcrafted semantic representation like dialogue act. This is an important step forward, not only because it allows the generator to produce more contextual and natural replies, but also because it requires less manual annotation labour and the complementary data collection procedure is much easier to scale using crowdsourcing platforms.

---

[9] The vanilla NDM was analysed here.

In the second section, we reviewed a series of studies on end-to-end dialogue modelling, including both chat-based and goal-oriented dialogue modelling. We concluded that a dialogue agent must be based on real-world knowledge or databases that enable causal and rational system behaviours. Furthermore, to develop systems that only have a limited access to training data, we introduce a coarse-grained intermediate representation so that the model can learn effectively on less training data by leveraging the intermediate annotations without compromising the system's scalability.

The main theme of this chapter is that we propose a Neural Dialogue Model for goal-oriented dialogue modelling by balancing the strengths and weaknesses of two dialogue research communities: firstly, the model is end-to-end trainable. However, it remains modularly connected. This allows us to combine the advantages of both end-to-end chat bots and modular task-oriented dialogue systems; secondly, it has an explicit representation of database (DB) attributes (slot-value pairs) which it uses to achieve high-task success rates, but has a distributed representation of user intent to allow ambiguous inputs; thirdly, it uses delexicalisation and a weight tying strategy (Henderson et al., 2014c) to reduce the data required to train the model, but still maintains a high degree of freedom should larger amounts of data become available. We show that the NDM can learn from a human-human conversational corpus via supervised learning and perform a task competitively across several metrics, when trained on only a few hundred dialogues.

Furthermore, to train the model for the target application, we introduced a novel pipelined data collection mechanism inspired by the Wizard-of-Oz paradigm (Kelley, 1984) to collect human-human dialogue corpora via crowd-sourcing. This data collection method collects dialogue corpora on a turn-by-turn basis and allows crowdsourcing to be easily done by non-experts. We found that this process is simple and enables fast data collection online with very low development costs.

Table 7.4 Samples of real text conversations between online judges and NDM.

Human	: Restaurant in any area that serves Vietnamese food
Machine	: thanh binh serves vietnamese food type , is there anything else i can help you with ?
Human	: What is the phone number and post code
Machine	: the number is 01223 362456 and the area code is c.b 3, 0 a.f . is there anything else i can do for you ?
Human	: Good bye
Machine	: thank you for using the cambridge restaurant system .
Human	: i am looking for moderately priced halal food
Machine	: i am sorry , there are no halal restaurants in the moderate price range . would you like something else ?
Human	: how about european food
Machine	: galleria serves european food type and is in the moderate price range .
Human	: address and postcode
Machine	: the address is 33 bridge street and the area code is c.b 2, 1 u.w . is there anything else i can help you with today ?
Human	: thanks
Machine	: galleria is a european restaurant in the moderate price range . would you like their phone ?
Human	: thank you
Machine	: thank you for using the cambridge restaurant system . good bye
Human	: Hi, I'm hungry for some Indonesian food
Machine	: i am sorry , there are no indonesian restaurants . would you like a different food ?
Human	: Oh no. How about Indian?
Machine	: there are 9 restaurants serving indian types of food . do you have a preference on side of town ?
Human	: East side please
Machine	: sitar tandoori is a indian restaurant in the east side of town . would you like their information ?
Human	: Yes, the address and phone number please
Machine	: sitar tandoori is located at 43 high street cherry hinton cherry hinton and their phone number is 01223 249955 .
Human	: Okay, thank you
Machine	: thank you , goodbye .
Human	: hello
Machine	: what cuisine would you like ?
Human	: afghan
Machine	: i am sorry , there are no afghan restaurants . would you like something else ?
Human	: korean
Machine	: little seoul serves korean food , it is located at 108 regent street city centre .
Human	: is it expensive?
Machine	: little seoul serves korean food and is in the expensive price range .
Human	: what area is that
Machine	: little seoul is located in the centre area of town . would you like their number ?
Human	: yes
Machine	: the telephone number is 01223 308681 . is there anything else i can help you with today ?
Human	: no goodbye
Machine	: thank you for using the cambridge restaurant system , goodbye .
Human	: I want to find a restaurant in the east part of town that serves creative food
Machine	: i am sorry , there are no creative restaurants in the east side of town . would you like a different restaurant ?
Human	: how about indian food?
Machine	: curry prince is a indian restaurant in the east of town . would you like their phone number ?
Human	: yes, may i have the address and phone number?
Machine	: curry prince is located at 451 newmarket road fen ditton and their phone is 01223 566388 . is there anything else i can help you with today ?
Human	: no, thank you goodbye
Machine	: thank you for using the cambridge restaurant system . good bye
Machine	: what food are you interested ?
Human	: english
Machine	: i am sorry , there are no english restaurants . would you like to try a different food type ?
Human	: chinese
Machine	: there are 8 restaurants serving chinese type of cuisine . do you have a preference of area ?
Human	: south part
Machine	: the good luck chinese food takeaway is a chinese restaurant in the south of town . would you like the phone number ?
Human	: no. just address and price range
Machine	: the good luck chinese food takeaway is located at 82 cherry hinton road cherry hinton . is there anything else i can help you with ?
Human	: what is their price range?
Machine	: the good luck chinese food takeaway is expensively priced chinese restaurant in the south area . would you like more information ?
Human	: no goodbye
Machine	: thank you for using the cambridge restaurant system . good bye .
Human	: Is there a gastropub anywhere around here?
Machine	: there are 3 gastropub restaurants . would you like more information ?
Human	: Is there one on the west side?
Machine	: i am sorry , there are no gastropub restaurants in the west side of town . would you like something else ?
Human	: How about east?
Machine	: royal standard is a gastropub restaurant in the east of town .
Human	: What's the address?
Machine	: the address is 290 mill road city centre .
Human	: Thanks, bye
Machine	: thank you for using the cambridge restaurant system . good bye

Chapter 8

Generation based on a Latent Policy

In the previous chapter, we proposed the idea of dialogue response generation by conditioning over the rich dialogue context instead of dialogue acts and learning end-to-end dialogue modelling directly, via decoder supervisions. Although the proposed NDM has demonstrated that it can learn conversational strategies from the training corpus, and performs quite well in achieving task success, NDM has an apparent defect. NDM cannot model intrinsic variations in both dialogue policy and language expressions. Like most neural networks such as LSTMs and sequence-to-sequence models, NDM also belongs to the family of deterministic models, where outcomes are precisely determined via known relationships among states and events, without room for random variation. Typically, discriminative models are trained to learn only conditional output distributions over strings, and despite the sophisticated architecture and conditioning mechanisms used to ensure salience, they cannot model the underlying actions needed to generate natural dialogues. Thus, these models are limited by their inability to exhibit the intrinsic variability and stochasticity of natural dialogue. In addition, there is often insufficient training data for goal-oriented dialogues, which results in over-fitting or preventing deterministic models from learning effective and scalable interactions. Another difficulty in end-to-end based dialogue models is the lack of an interpretable interface for controlling system responses, which poses a great challenge when debugging of real-world applications.

In this chapter, we propose to address NDM's drawbacks with a latent variable model and introduce the Latent Intention Dialogue Model (LIDM), that can learn complex distributions of communicative intentions in goal-oriented dialogues. In LIDM, each system action is represented as a dimension of a discrete latent variable, instead of a deterministic vector as in NDM. The system action (or intention), which is sampled from the latent distribution that was inferred from the user's input utterance, is combined with the dialogue context representation that will guide the generation of the system response. Since the learnt latent space is discrete, it can be interpreted by the responses it generates; therefore, the space serves as an interface for human intervention and manipulation. Furthermore, this discrete latent variable also provides the model with an interface to perform different learning paradigms under the same framework. This is important because it provides a stepping stone towards building an autonomous dialogue agent that can continuously improve itself via interaction.

8.1 Stochastic Policy as a Discrete Latent Variable

As mentioned in the beginning of this chapter, if we view end-to-end dialogue modelling as a single problem, then latent variable modelling of the communicative intention must capture the underlying diversity and variation in the conversation.

To model intention as a latent variable, we must first decide whether we want to model it as a continuous or discrete latent variable. Continuous latent variable modelling is more popular and has been introduced in chat-based scenarios before (Cao and Clark, 2017; Serban et al., 2016). The chat-based scenario of a continuous Gaussian random variable is sampled and the Gaussian re-parameterisation trick is used for inference. Although this continuous Gaussian random variable injects randomness and encourages diversity in the response, it does not bring the additional advantage, like the discrete latent variables do. In this thesis, we propose to model the dialogue intention as a discrete latent variable instead.

There are several benefits to modelling dialogue intention as a discrete latent variable (a categorical distribution) rather than a continuous one (a Gaussian distribution): firstly, each dimension of the discrete latent distribution can be interpreted as a DA and the distribution can be considered a latent dialogue policy. Unlike the POMDP-based framework (Young et al., 2013), which handcrafted the set of system intentions based on the DA formalism (Traum, 1999), here we directly infer the underlying dialogue intentions from data. Thus, we can handle intent distributions with long tails by measuring similarities, against the existing ones, during variational inference. This frees the model from pre-defined, limited semantic representations and allows it to better capture complex human-human conversations; secondly, this discrete set of intentions can serve as a principal framework for integrating different learning paradigms. This allows us to develop a dialogue agent with an AlphaGo-like recipe (Silver et al., 2016) by first training on a supervised dataset and then applying reinforcement learning to continuously improve through interaction. This discrete interface is also good for semi-supervised learning, which is critical for NLP tasks where additional supervision and external knowledge can be utilised for bootstrapping (Faruqui et al., 2015; Kočiský et al., 2016; Miao and Blunsom, 2016); finally, this discrete latent variable can also serve as an interface for human control and intervention. This is crucial from a practical viewpoint because it not only allows us to better understand the complex learning system, but also provides us with a gateway for debugging it.

8.2 Latent Intention Dialogue Model

The Latent Intention Dialogue Model (LIDM) proposed here is based on the framework of NDM as described in the previous chapter. However, instead of modelling each system action as a deterministic vector, we introduce a discrete latent variable with a categorical distribution to better capture its underlying stochasticity.

In LIDM, the latent intention is inferred from user input utterances. The agent draws a sample based on its dialogue context. The intention then guides the generation of natural language response. Firstly, in the framework of NVI (Mnih and Gregor, 2014), we construct an inference network to approximate the posterior distribution over the latent intention. Then, by sampling the intentions for each response, we can directly learn a basic intention distribution on a human-human dialogue corpus by optimising the variational lower bound. To further reduce the variance, we utilise a labelled subset of the corpus. The labels are automatically generated by clustering. Then, the latent intention distribution can be learned in a semi-supervised fashion, where the learning signals are either from the direct supervision (labelled set) or the variational lower bound (unlabelled set).

From the perspective of reinforcement learning, the latent intention distribution can be interpreted as the intrinsic policy that reflects human decision-making under a conversational scenario. Based on the initial policy (latent intention distribution) learnt from the semi-supervised variational inference framework, the model can refine its strategy easily against alternative objectives, especialy by using policy gradient-based reinforcement learning. This is somewhat analogous to the training process used in AlphaGo (Silver et al., 2016) for the game of Go. Based on LIDM, we show that different learning paradigms can be brought together under the same framework to bootstrap the development of a dialogue agent (Li et al., 2016c; Weston, 2016).

8.2.1 Model

The LIDM is based on the NDM framework proposed in Chapter 7. For the sake of explaining NVI in the dialogue system framework, we give an alternative modular interpretation of the system diagram that is shown in Figure 8.1, which comprises three principal components: (1) Representation Construction; (2) Policy Network; and (3) Generator.

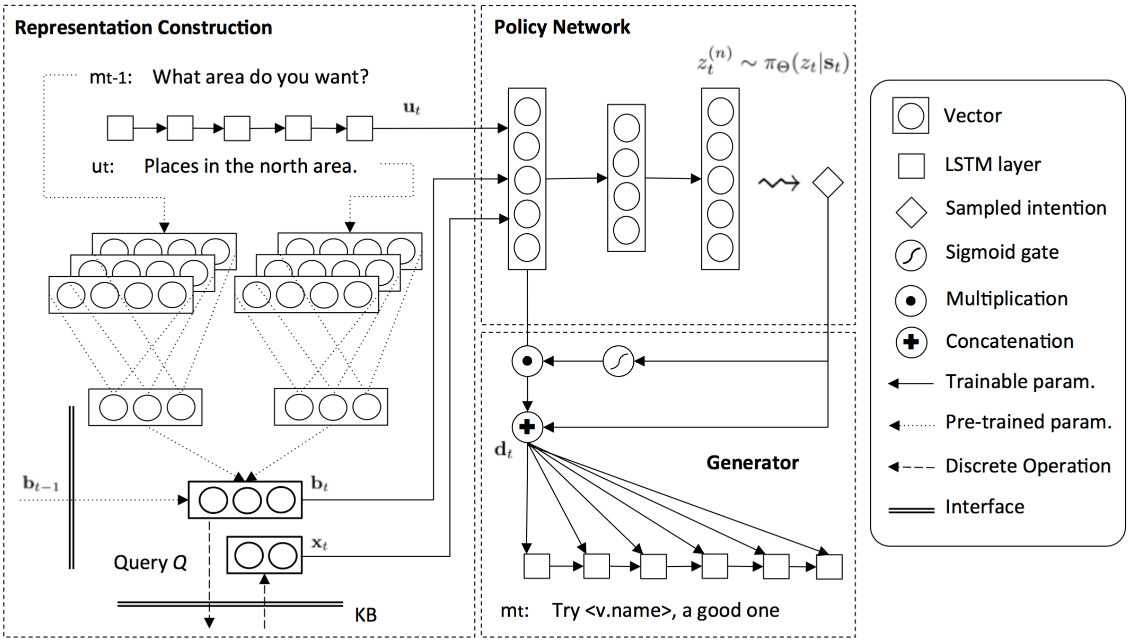


Fig. 8.1 LIDM for Goal-oriented Dialogue Modelling

Representation Construction Based on the user input query $\mathrm{u_t}$ and the internal knowledge base KB, we first construct a representation of the dialogue context via

$$\mathbf{s}_t = \mathbf{u}_t \oplus \hat{\mathbf{b}}_t \oplus \mathbf{x}_t. \tag{8.1}$$

As we recall from Section 7.3, $\mathbf{u}_\mathrm{t}$ is a bidirectional LSTM encoding of the user input utterance $\mathbf{u}_\mathrm{t} = \mathrm{biLSTM}_\theta(\mathrm{u_t})$, and $\hat{\mathbf{b}}_\mathrm{t}$ is a concatenation of a set of summarised probability distributions over domain specific slot-value pair. This set is extracted by a set of pre-trained RNN-CNN belief trackers (Mrkšić et al., 2016b; Wen et al., 2017b), in which $\mathrm{u_t}$ and $\mathrm{m_{t-1}}$ are processed by two different CNNs as shown in Figure 8.1

$$\mathbf{b}_t = \text{RNN-CNN}(u_t, m_{t-1}, \mathbf{b}_{t-1}), \tag{8.2}$$

where $\mathrm{m_{t-1}}$ is the preceding machine response and $\mathbf{b}_{\mathrm{t}-1}$ is the preceding belief vector. They are included to model the current turn of the discourse and the long-term dialogue context, respectively. Based on the belief vector, a query $\mathrm{q_t}$ is formed by taking the union of the maximum values of each slot. $\mathrm{q_t}$ is then used to search the internal KB and return a vector $\mathbf{x}_\mathrm{t}$, which represents the degree of matching in the KB. This is produced by counting all the matching venues and re-structuring it into a six-bin one-hot vector. Up to now, we reuse NDM's system modules to construct the deterministic dialogue state vector $\mathbf{s}_\mathrm{t}$.

Latent Policy Network Conditioning on the state $\mathbf{s}_t$, the policy network parameterises the latent intention z_t via a single layer MLP

$$\pi_\theta(z_t|\mathbf{s}_t) = \mathrm{softmax}(\mathbf{W}_2^\intercal \cdot \tanh(\mathbf{W}_1^\intercal \mathbf{s}_t + \mathbf{b}_1) + \mathbf{b}_2), \tag{8.3}$$

where $\mathbf{W}_1$, $\mathbf{b}_1$, $\mathbf{W}_2$, $\mathbf{b}_2$ are model parameters. Since $\pi_\theta(z_t|\mathbf{s}_t)$ is a discrete conditional probability distribution, based on dialogue state, we can also interpret the policy network as a latent dialogue management component as in the traditional POMDP-based framework (Gašić et al., 2013; Young et al., 2013). A latent intention $z_t^{(n)}$ (or an action in the reinforcement learning literature) can then be sampled from the conditional distribution

$$z_t^{(n)} \sim \pi_\theta(z_t|\mathbf{s}_t). \tag{8.4}$$

Generator The sampled intention (or action) $z_t^{(n)}$ and the state vector $\mathbf{s}_t$ can be combined into a control vector $\mathbf{d}_t$, which is then used to govern the generation of the system response m_t, based on a conditional LSTM language model

$$\mathbf{d}_t = \mathbf{W}_4^\intercal \mathbf{z}_t \oplus \left[\mathrm{sigmoid}(\mathbf{W}_3^\intercal \mathbf{z}_t + \mathbf{b}_3) \cdot \mathbf{W}_5^\intercal \mathbf{s}_t\right] \tag{8.5}$$

$$p_\theta(m_t|\mathbf{s}_t, z_t) = \prod_j p(w_{j+1}^t | w_j^t, \mathbf{h}_{j-1}^t, \mathbf{d}_t), \tag{8.6}$$

where $\mathbf{b}_3$ and $\mathbf{W}_{3\sim5}$ are parameters, $\mathbf{z}_t$ is the 1-hot representation of $z_t^{(n)}$, w_j^t is the last output token (i.e., a word, a delexicalised slot name or a delexicalised slot value), and $\mathbf{h}_{j-1}^t$ is the decoder's last hidden state. Note in Equation 8.5, the degree of information flow from the state vector is controlled by a sigmoid gate whose input signal is the sampled intention $z_t^{(n)}$. This prevents the decoder from over-fitting to the deterministic state's information and forces it to consider the sampled stochastic intention. The LIDM can then be formally written down by its parameterised form with a parameter set θ,

$$p_\theta(m_t|\mathbf{s}_t) = \sum_{z_t} p_\theta(m_t|z_t, \mathbf{s}_t)\pi_\theta(z_t|\mathbf{s}_t). \tag{8.7}$$

8.2.2 Inference

To carry out inference for LIDM, we introduce an inference network $q_\phi(z_t|\mathbf{s}_t, m_t)$ to approximate the posterior distribution $p(z_t|\mathbf{s}_t, m_t)$. We then optimise the variational lower bound of the joint probability in the neural variational inference framework (Miao et al., 2016) and thus, derive the variational lower bound

$$\begin{aligned}\mathcal{L}(\theta,\phi) &= \mathbb{E}_{\mathrm{q}_\phi(\mathrm{z_t})}[\log \mathrm{p}_\theta(\mathrm{m_t}|\mathrm{z_t},\mathbf{s}_\mathrm{t})] - \lambda \mathrm{D}_\mathrm{KL}(\mathrm{q}_\phi(\mathrm{z_t})||\pi_\theta(\mathrm{z_t}|\mathbf{s}_\mathrm{t})) \\ &\le \log \sum_{\mathrm{z_t}} \mathrm{p}_\theta(\mathrm{m_t}|\mathrm{z_t},\mathbf{s}_\mathrm{t})\pi_\theta(\mathrm{z_t}|\mathbf{s}_\mathrm{t}) \\ &= \log \mathrm{p}_\theta(\mathrm{m_t}|\mathbf{s}_\mathrm{t}),\end{aligned} \tag{8.8}$$

where $\mathrm{q}_\phi(\mathrm{z_t})$ is shorthand for $\mathrm{q}_\phi(\mathrm{z_t}|\mathbf{s}_\mathrm{t},\mathrm{m_t})$. Note that we use the modified version of the lower-bound here by introducing the trade-off constant λ (Higgins et al., 2017). By following the steps, as described in Section 4.3.2, the inference network $\mathrm{q}_\phi(\mathrm{z_t}|\mathbf{s}_\mathrm{t},\mathrm{m_t})$ is then constructed by

$$q_\phi(z_t|\mathbf{s}_t,m_t) = \mathrm{Multi}(\mathbf{o}_t) = \mathrm{softmax}(\mathbf{W}_6\mathbf{o}_t) \tag{8.9}$$

$$\mathbf{o}_t = \mathrm{MLP}_\phi(\hat{\mathbf{b}}_t,\mathbf{x}_t,\mathbf{u}_t,\mathbf{m}_t) \tag{8.10}$$

$$\mathbf{u}_t = \mathrm{biLSTM}_\phi(u_t), \mathbf{m}_t = \mathrm{biLSTM}_\phi(m_t), \tag{8.11}$$

where $\mathbf{o}_\mathrm{t}$ is the joint representation, and both $\mathbf{u}_\mathrm{t}$ and $\mathbf{m}_\mathrm{t}$ are modelled by a bidirectional LSTM network. Although both $\mathrm{q}_\phi(\mathrm{z_t}|\mathbf{s}_\mathrm{t},\mathrm{m_t})$ and $\pi_\theta(\mathrm{z_t}|\mathbf{s}_\mathrm{t})$ are modelled as parameterised multinomial distributions, $\mathrm{q}_\phi(\mathrm{z_t}|\mathbf{s}_\mathrm{t},\mathrm{m_t})$ is an approximation that only functions via inference by producing samples to compute stochastic gradients, while $\pi_\theta(\mathrm{z_t}|\mathbf{s}_\mathrm{t})$ is the generative distribution that generates the samples for composing the machine response.

Based on the samples $\mathrm{z_t^{(n)}} \sim \mathrm{q}_\phi(\mathrm{z_t}|\mathbf{s}_\mathrm{t},\mathrm{m_t})$, we adopt a scoring function-based gradient estimator, as described in Section 4.3.2. We use different strategies to alternately optimise the parameters θ and ϕ against the variational lower bound (Equation 8.8). To optimise parameters θ, we further divide the parameter set into two sets of parameters $\theta = \{\theta_1,\theta_2\}$. For θ_1 on the decoder side, we directly update them by back-propagating the gradients,

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial \theta_1} &= \mathbb{E}_{\mathrm{q}_\phi(\mathrm{z_t}|\mathbf{s}_\mathrm{t},\mathrm{m_t})}[\frac{\partial \log \mathrm{p}_{\theta_1}(\mathrm{m_t}|\mathrm{z_t},\mathbf{s}_\mathrm{t})}{\partial \theta_1}] \\ &\approx \frac{1}{\mathrm{N}}\sum_{\mathrm{n}} \frac{\partial \log \mathrm{p}_{\theta_1}(\mathrm{m_t}|\mathrm{z_t^{(n)}},\mathbf{s}_\mathrm{t})}{\partial \theta_1}\end{aligned} \tag{8.12}$$

For the parameters θ_2 in the generative network, we update them by minimising the KL divergence via

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial \theta_2} &= -\frac{\partial \lambda \mathrm{D_{KL}}(\mathrm{q}_\phi(\mathrm{z_t}|\mathbf{s}_\mathrm{t},\mathrm{m_t})||\pi_{\theta_2}(\mathrm{z_t}|\mathbf{s}_\mathrm{t}))}{\partial \theta_2} \\ &= -\lambda \sum_{\mathrm{z_t}} \mathrm{q}_\phi(\mathrm{z_t}|\mathbf{s}_\mathrm{t},\mathrm{m_t}) \frac{\partial \log \pi_{\theta_2}(\mathrm{z_t}|\mathbf{s}_\mathrm{t})}{\partial \theta_2}\end{aligned} \tag{8.13}$$

The entropy derivative $\partial \mathrm{H}[\mathrm{q}_\phi(\mathrm{z_t}|\mathbf{s}_\mathrm{t},\mathrm{m_t})]/\partial \theta_2 = 0$; thus, it can be ignored. Finally, for each parameter ϕ in the inference network, we define the learning signal $\mathrm{r}(\mathrm{m_t},\mathrm{z_t^{(n)}},\mathbf{s}_\mathrm{t})$ as

$$\mathrm{r}(\mathrm{m_t},\mathrm{z_t^{(n)}},\mathbf{s}_\mathrm{t}) = \log \mathrm{p}_{\theta_1}(\mathrm{m_t}|\mathrm{z_t^{(n)}},\mathbf{s}_\mathrm{t}) - \lambda(\log \mathrm{q}_\phi(\mathrm{z_t^{(n)}}|\mathbf{s}_\mathrm{t},\mathrm{m_t}) - \log \pi_{\theta_2}(\mathrm{z_t^{(n)}}|\mathbf{s}_\mathrm{t})). \tag{8.14}$$

Then the ϕ parameter can be updated by using

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial \phi} &= \mathbb{E}_{\mathrm{q}_\phi(\mathrm{a_t}|\mathbf{s}_\mathrm{t},\mathrm{m_t})}[\mathrm{r}(\mathrm{m_t},\mathrm{a_t},\mathbf{s}_\mathrm{t}) \frac{\partial \log \mathrm{q}_\phi(\mathrm{a_t}|\mathbf{s}_\mathrm{t},\mathrm{m_t})}{\partial \phi}] \\ &\approx \frac{1}{\mathrm{N}} \sum_{\mathrm{n}} \mathrm{r}(\mathrm{m_t},\mathrm{z_t^{(n)}},\mathbf{s}_\mathrm{t}) \frac{\partial \log \mathrm{q}_\phi(\mathrm{z_t^{(n)}}|\mathbf{s}_\mathrm{t},\mathrm{m_t})}{\partial \phi}\end{aligned} \tag{8.15}$$

This gradient estimator has a large variance because the learning signal $\mathrm{r}(\mathrm{m_t},\mathrm{z_t^{(n)}},\mathbf{s}_\mathrm{t})$ relies on samples from the proposal distribution $\mathrm{q}_\phi(\mathrm{z_t}|\mathbf{s}_\mathrm{t},\mathrm{m_t})$. To reduce the variance during inference, we follow the REINFORCE algorithm (Mnih and Gregor, 2014; Mnih et al., 2014) to introduce the two baselines b and $\mathrm{b}(\mathbf{s}_\mathrm{t})$. The centred baseline signal and input dependent baseline help reduce the variance. b is a learnable constant and $\mathrm{b}(\mathbf{s}_\mathrm{t}) = \mathrm{MLP}(\mathbf{s}_\mathrm{t})$ is modelled by an MLP. During training, the two baselines are updated by minimising the distance,

$$\mathcal{L}_b = \Big[r(m_t,z_t^{(n)},\mathbf{s}_t) - b - b(\mathbf{s}_t)\Big]^2 \tag{8.16}$$

and the gradient w.r.t. ϕ can be rewritten as

$$\frac{\partial \mathcal{L}}{\partial \phi} \approx \frac{1}{N} \sum_n [r(m_t,z_t^{(n)},\mathbf{s}_t) - b - b(\mathbf{s}_t)] \frac{\partial \log q_\phi(z_t^{(n)}|\mathbf{s}_t,m_t)}{\partial \phi} \tag{8.17}$$

8.2.3 Semi-supervised Learning

Despite the efforts to reduce the variance, there are two major difficulties in learning latent intentions in a completely unsupervised manner: (1) the high variance of the inference network in the early stages of training prevents it from generating sensible intention samples in the beginning, and (2) The overly-strong discriminative power of the LSTM language model. This results in a *disconnection* phenomenon between the LSTM decoder and the rest of the system components. The decoder learns to ignore the samples but only to focus on optimising the language model. To ensure a more stable training experience and prevent disconnection, a semi-supervised learning technique is introduced.

Inferring the latent intentions behind utterances is like an unsupervised clustering task. To mitigate the learning load of the model, we can use off-the-shelf clustering algorithms to pre-process the corpus and generate automatic labels $\hat{z}_t$ for part of the training examples $(m_t, \mathbf{s}_t, \hat{z}_t) \in \mathbb{L}$. As a result, when the model is trained on the unlabelled examples $(m_t, \mathbf{s}_t) \in \mathbb{U}$, we optimise the model against the modified variational lower bound as in Equation 8.8.

$$\mathcal{L}_1 = \sum_{(m_t, \mathbf{s}_t) \in \mathbb{U}} \mathbb{E}_{q_\phi(z_t|\mathbf{s}_t, m_t)} [\log p_\theta(m_t|z_t, \mathbf{s}_t)] - \lambda D_{KL}(q_\phi(z_t|\mathbf{s}_t, m_t)||\pi_\theta(z_t|\mathbf{s}_t)) \tag{8.18}$$

If the model is updated based on examples from the labelled set $(m_t, \mathbf{s}_t, \hat{z}_t) \in \mathbb{L}$, we treat the labelled intention $\hat{z}_t$ as an observed variable and train the model by maximising the joint log-likelihood,

$$\mathcal{L}_2 = \sum_{(m_t, \hat{z}_t, \mathbf{s}_t) \in \mathbb{L}} \log \left[p_\theta(m_t|\hat{z}_t, \mathbf{s}_t) \pi_\theta(\hat{z}_t|\mathbf{s}_t) q_\phi(\hat{z}_t|\mathbf{s}_t, m_t) \right] \tag{8.19}$$

Thus, the final joint objective function can be written as $\mathcal{L}' = \alpha\mathcal{L}_1 + \mathcal{L}_2$, where α is a trade-off constant on the supervised and unsupervised examples.

8.2.4 Reinforcement Learning

One of the main purposes of learning interpretable, discrete latent intention within a dialogue system is to be able to control and refine the model's future behaviour. The generative network $\pi_\theta(z_t|\mathbf{s}_t)$ encodes the policy discovered from the underlying data distribution, but it is not necessarily optimal for any task. Since the $\pi_\theta(z_t|\mathbf{s}_t)$ is a parameterised policy network itself, any off-the-shelf, policy gradient-based reinforcement learning algorithm (Konda and Tsitsiklis, 2003; Williams, 1992) can be used to fine-tune the initial policy against other objective functions that we are more interested in.

Based on the initial policy $\pi_\theta(z_t|\mathbf{s}_t)$, we revisit the training dialogues and update parameters based on the following strategy: when encountering unlabelled examples $\mathbb{U}$, in turn t, the system samples an action from the learnt policy $z_t^{(n)} \sim \pi_\theta(z_t|\mathbf{s}_t)$ and receives a reward $r_t^{(n)}$. By conditioning on these, we can directly fine-tune a subset of the model parameters θ' via the policy gradient method

$$\frac{\partial \mathcal{J}}{\partial \theta'} \approx \frac{1}{N} \sum_n r_t^{(n)} \frac{\partial \log \pi_\theta(z_t^{(n)}|\mathbf{s}_t)}{\partial \theta'}, \tag{8.20}$$

where $\theta' = \{\mathbf{W}_1, \mathbf{b}_1, \mathbf{W}_2, \mathbf{b}_2\}$ is the MLP that parameterises the policy network (Equation 8.3). When a labelled example $\in \mathbb{L}$ is encountered, we force the model to take the labelled action $z_t^{(n)} = \hat{z}_t$ and update the parameters via Equation 8.20. Unlike Li et al. (2016c), which refines the whole model end-to-end using RL, updating only θ' effectively allows us to revise only the *decision-making* of the system and make it resist against over-fitting.

8.3 Evaluation on Latent Intention Dialogue Models

8.3.1 Experimental Setup

We tested the LIDM using the CamRest676 corpus, as described in Section 7.4. To make a direct comparison with NDMs, we follow the same experimental setup as in Section 7.5. For more detail about the common experimental settings and the hyper-parameters, please refer back to Section 7.5.

For LIDM specific settings, we train three types of LIDMs with their latent intention size I set to 50, 70, and 100, respectively. The trade-off constants λ and α are both empirically set to 0.1. To produce self-labelled response clusters for semi-supervising the intentions, we firstly removed function words from all the responses and cluster them according to their word content. We then assign responses to the i-th frequent cluster to the i-th latent dimension as its supervised set. This results in about 35% ($I = 50$) to 43% ($I = 100$)-labelled responses across the whole dataset. An example of the resulting seed set is shown in Table 8.1. This seed set is somehow related to the DA taxonomies used in this domain, but instead of labeling their semantics manually, in LIDM we only roughly cluster than based their surface form content words. During inference, we carried out stochastic estimation by taking a sample that estimates the stochastic gradients. The model is trained by Adam (Kingma and Ba, 2014) and tuned (early stopping, hyper-parameters) on the held-out validation set. We alternately optimise the generative model and the inference network by fixing the parameters of one, while updating the parameters of the other.

Due to the difficulty in training end-to-end models directly online with real users, a rather simple corpus-based RL setting was adopted to prove the validity of applying RL on top of the trained latent variable. During corpus RL fine-tuning, we generate a sentence m_t from the model to replace the ground truth $\hat{m}_t$ at each turn and define an immediate reward. If m_t

ID	#	content words
0	138	thank, goodbye
1	91	welcome, goodbye
3	42	phone, address, [v.phone],[v.address]
14	17	address, [v.address]
31	9	located, area, [v.area]
34	9	area, would, like
46	7	food, serving, restaurant, [v.food]
85	4	help, anything, else

Table 8.1 An example of the automatically labelled response seed set for semi-supervised learning during variational inference.

can improve the dialogue success Su et al. (2015) over $\hat{\mathrm{m}}_\mathrm{t}$ and the sentence BLEU (Auli and Gao, 2014) via

$$\mathrm{r_t} = \eta \cdot \mathrm{sBLEU}(\mathrm{m_t}, \hat{\mathrm{m}}_\mathrm{t}) + \begin{cases} 1 & \mathrm{m_t} \text{ improves,} \\ -1 & \mathrm{m_t} \text{ degrades,} \\ 0 & \text{otherwise,} \end{cases} \tag{8.21}$$

where the constant η was set to 0.5. We fine-tuned the model parameters using RL for 3 epochs. During testing, we selected the most probable intention using a greedy method and applied a beam search with the beamwidth set to 10 when decoding the response. The decoding criterion was the average log-probability of tokens in the response. We then evaluated our model based on task success rate (Su et al., 2015) and BLEU score (Papineni et al., 2002). The model is used to predict each system response in the held-out test set.

8.3.2 Results

Like the results presented in Section 7.5.2, we ran a small grid search over the same set of hyper-parameters for LIDM. Thus, the initial learning rate of Adam whose range was $[0.002, 0.004, 0.006]^1$; the l2-regularisation term had range of $[0.0, 1e-6, 1e-5, 1e-4]$; and the random seed values range from 1 to 5. We trained a model based on each of the hyper-parameter combinations mentioned above. We picked the model, which performed the best on the validation set, and reported its performance on the testing set. Note the learning rate range searched here is smaller than that of the NDM's because NVI requires a smaller

---

[1] For NDM we searched over [0.008,0.010,0.012]

Table 8.2 Corpus-based experiment by comparing different NDM and LIDM architectures. The results were obtained by training models with several hyper-parameter settings and report the testing performance based on the best model on the validation set. The best performance of each metric in each big block is highlighted in **bold**.

Model	Decoder	Addition	Success(%)	BLEU	Suc.+0.5 BLEU
Vanilla NDM	lm	-	72.8%	0.237	0.847
	mem	-	74.3%	0.243	0.865
	hybrid	-	**77.9%**	0.231	**0.894**
Attentive NDM	lm	-	72.1%	0.246	0.844
	mem	-	**80.1%**	0.240	**0.921**
	hybrid	-	77.9%	0.234	0.896
Model	Intent Dim.	Addition	Success(%)	BLEU	Suc.+0.5 BLEU
LIDM	I=50	-	78.7%	0.226	0.900
	I=70	-	80.9%	0.245	0.932
	I=100	-	69.1%	0.221	0.801
	I=50	+RL	77.2%	0.249	0.896
	I=70	+RL	**83.8%**	**0.258**	**0.967**
	I=100	+RL	81.6%	0.245	0.939
Ground Truth	-	-	91.6%	1.000	1.416

initial learning rate to explore the latent space in the first few epochs. The corpus-based evaluation results are presented in Table 8.2. Note that the result of NDMs (first two blocks of Table 8.2) are borrowed from Table 7.1 for the convenience of comparison.

As seen in Table 8.2, the *Ground Truth* block shows the two metrics when we computed them on the human-authored responses. This sets a gold standard for the task. When comparing among LIDMs (the LIDM block), a dimension of 70 satisfies the number of latent intentions on this corpus because both the success rate and BLEU rate are the best among others. Furthermore, the initial policy learnt by fitting the latent intention to the underlying data distribution does not necessarily produce good results when compared to both the vanilla and attentive NDMs. This may be because we were optimising the variational lower bound of the dataset instead of the task success and BLEU score during the variational inference. However, once we applied RL to optimise the success rate and BLEU rate as part of the reward function (Equation 8.21) during the fine-tuning phase, the resulting RL-based LIDM models can outperform all NDM baselines when both the success rate and BLEU score are considered. To observe the general trend of the three types of models, we highlighted the best performance of each metric (each column) on each model (each big block) in bold. As

Metrics	NDM	LIDM	LIDM+RL
Success	91.5%	92.0%	93.0%
Comprehension	4.21	4.40	4.40
Naturalness	4.08	4.29	4.28
# of Turns	4.45	4.54	4.29

Table 8.3 Human evaluation based on text-based conversations.

we can see, the best metrics of each model suggests that the proposed LIDM performs better in terms of all the metrics considered.

8.3.3 Human Evaluation

To assess the human perceived performance, we evaluated the three models (1) Attentive NDM, (2) LIDM (without RL), and (3) LIDM+RL by recruiting paid subjects on Amazon Mechanical Turk. Each judge was asked to follow a task and carry out a conversation with the machine. At the end of the conversation the judges were asked to rate and compare the model's performance. We assessed the subjective success rate, the perceived comprehension ability and the naturalness of responses on a scale of 1 to 5. For each model, we collected 200 dialogues and averaged the resulting scores. During the human evaluation, we sampled from the top-5 intentions of the LIDM models and decoded a response based on the sample. The result is shown in Table 8.3. One interesting fact to note is that although the LIDM did not perform well on the corpus-based task success metric, the human judges rated its subjective success almost indistinguishable from the others. This discrepancy between the two experiments suggests that the flaw in the corpus-based success metric that favours greedy policies because the user side behaviours were fixed rather than interactional[2]. Despite LIDMs being considered only marginally better than NDM on the subject of success, the LIDMs outperform NDM on both comprehension and naturalness scores. This is because the proposed LIDM models can better capture the multi-modality of the communicative intention in dialogue and carry out a more natural conversation by sampling from the latent variable.

In a qualitative experiment, we show three example conversations between a human judge and a machine, one from LIDM in Table 8.4 and two from LIDM+RL in Table 8.5, respectively. The results are shown as one exchange per block. Each induced latent intention

---

[2] The system wants to provide as much information as possible in early turns, in case that the fixed user side behaviours a few turns later don't fit into the scenario the system originally planned. A simulated user is impossible here because LIDM directly operates at the language level and developing an end-to-end simulated user itself is an AI-complete problem

Human :	Hi, I'm hungry for some Indonesian	
Machine:	**(0 0.71)**	**i am sorry , there are no indonesian restaurants in the area . would you like something else ?**
	(46 0.19)	there are no restaurants serving indonesian kind of food .
	(34 0.07)	what location would you like ?
Human :	Oh no. How about Indian?	
Machine:	(0 0.59)	there are 9 restaurants in the east side of town . would you like their address ?
	(30 0.15)	curry prince serves indian cuisine .
	(34 0.10)	**what area would you like ?**
	(16 0.09)	curry prince serves indian food , is in the east part of town .
Human :	East side please	
Machine:	**(0 0.62)**	**curry prince is a indian restaurant in the east side of town . would you like their telephone number ?**
	(84 0.19)	curry prince is an indian restaurant in the east of town and the number is 01223 566388 .
	(81 0.08)	curry prince serves indian type of cuisine and is located in the east part of town .
Human :	yes , the address and phone number please	
Machine:	(3 0.46)	the address is 451 newmarket road fen ditton and the phone is 01223 566388 .
	(11 0.16)	**the address for curry prince is 451 newmarket road fen ditton and the phone number is 01223 566388 .**
	(0 0.15)	curry prince is located at 451 newmarket road fen ditton and their phone is 01223 566388 . is there anything else i can help you with ?
	(84 0.13)	curry prince is located at 451 newmarket road fen ditton and the phone is 01223 566388 and is in the moderate price range .
Human :	Okay, thank you goodbye	
Machine:	(0 0.57)	thank you , goodbye .
	(2 0.13)	goodbye .
	(1 0.11)	**you are welcome . goodbye .**
	(6 0.10)	thank you for using the cambridge restaurant system . goodbye .

Table 8.4 A sample dialogue from the LIDM, I=100 model, one exchange per block. Each induced latent intention is shown using an (index, probability) tuple, followed by a decoded response. The sample dialogue was produced by following the responses highlighted in bold.

is shown via an (index, probability) tuple followed by a decoded response. The sample dialogues were produced by following the responses highlighted in bold. As we can see, the LIDM shown in Table 8.4 demonstrates a clear multi-modal distribution over the learnt intention latent variable, and what the variable represents can be easily interpreted by the response generated. Note the variation of the latent variable is at both the semantic and syntactic levels. However, some intentions (such as *intent 0*) can still decode very different responses under different dialogue states, even though they were supervised by a small response set as shown in Table 8.1. This is mainly because of the variance introduced during the variational inference. Finally, when comparing Table 8.4 and Table 8.5, we can observe the difference between the two dialogue strategies: the LIDM, by inferring its policy from the supervised dataset, reflects the rich multi-modality in the underlying distribution; while LIDM+RL, which refines its strategy using RL, exhibits much greedier behaviour for achieving task success (*Table 8.5 in block 2 & 4* the LIDM+RL agent provides the address and phone number even before the user asks). This greedy behaviour is mainly caused by two reasons: (1) the sparse nature of the success reward signal, and (2) the fixed user behaviour in the corpus-based RL setting [2]. This is also supported by the human evaluation in Table 8.3 where LIDM+RL has a much shorter dialogue turn in average when comparing to the other

Human :	Hi, I'm hungry for some Indonesian .		
Machine:	**(0**	**0.61)**	**i am sorry , there are no indonesian restaurants in the area . would you like something else ?**
	(46	0.17)	no restaurants in the part of town .
	(34	0.14)	what part of town would you like ?
Human :	Oh no. How about Indian?		
Machine:	**(93**	**0.51)**	**the gandhi is located at 72 regent street city centre and their telephone number is 01223 353942 .**
	(72	0.21)	the gandhi serves indian kind of food and is in the centre of town and is in the cheap price range .
	(12	0.12)	the gandhi is a indian restaurant in the centre location and is in the cheap price range .
Human :	Okay, thank you goodbye		
Machine:	**(6**	**0.28)**	**thank you for using the cambridge restaurant system . goodbye .**
	(1	0.24)	you are welcome . goodbye .
	(85	0.19)	is there anything else i can help you with ?
Human :	hey can i have some chinese cruisine in the centre ? and a cheap one please		
Machine:	(15	0.42)	golden house is located at 12 lensfield road city centre and their number is 01842 753771 .
	(5	**0.38)**	**golden house is located at 12 lensfield road city centre and the number is 01842 753771 . is there ...**
	(93	0.10)	golden house is located at 12 lensfield road city centre and their phone is 01842 753771 .
Human :	that is all		
Machine:	**(6**	**0.57)**	**thank you for using the cambridge restaurant system . goodbye .**
	(1	0.20)	you are welcome . goodbye .
	(17	0.11)	have a nice day !

Table 8.5 Two sample dialogues from the LIDM+RL, I=100 model, one exchange per block. Comparing to Table 8.4, the RL agent demonstrates a much greedier behavior toward task success. This can be seen in *block 2 & block 4* in which the agent provides the address and phone number even before the user asks.

two models. A better model can be achieved by learning from real user interaction with a more expressive reward signal.

8.4 Conclusions

In this chapter, we continued our study of end-to-end neural network-based dialogue models by introducing a discrete latent variable as its internal decision-making component. The discrete latent variable enables the model to make conversational decisions by sampling from a categorical distribution, which is inferred from a human-human dialogue corpus. The corpus can be interpreted as the intrinsic policy that reflects human decision-making under a particular conversational scenario.

The proposed LIDM is based on the framework of Conditional VAE as illustrated in Section 4.3.1. We introduce a discrete latent variable between the input and output nodes and optimise the model parameters against the variational lower bound. To carry out efficient neural variational inference, we construct an inference network that approximates the true posterior by encoding both the input and output as observations. In Section 4.3.2, we opt for the scoring function-based gradient estimator to optimise the model parameters via stochastic back-propagation. During NVI, we alternately optimise the generative model and the inference network by fixing the parameters of one, while updating the parameters of the other. To reduce the variance, baselines are also introduced, as in the REINFORCE algorithm, to mitigate the high-variance problem.

To further reduce the variance, we utilise a labelled subset of the corpus. The labels of intentions are automatically generated by clustering in this subset. Then, the latent intention distribution can be learned in a semi-supervised fashion. Thus, the learning signals are either from the direct supervision (labelled set) or from the variational lower bound (unlabelled set). Based on the same framework, we show that the model can refine its strategy easily against alternative objectives by using policy gradient-based reinforcement learning.

From a machine learning perspective, LIDM is attractive because the latent variable can serve as an interface. The interface, which is also learnt, can be used to decompose the learning of language and the internal dialogue decision-making process. This decomposition can effectively help us to resolve the credit assignment problem in end-to-end learning dialogue models. This is because different learning signals can be efficiently assigned to a variety of system sub-modules to update parameters. Variational inference, for discrete latent variables, uses a latent distribution that is updated by a reward from a variational lower bound. Reinforcement learning uses a latent distribution (i.e., policy network) that is updated by the rewards from dialogue success and sentence BLEU. Hence, the latent variable bridges the different learning paradigms such as Bayesian learning and Reinforcement Learning and brings them together under the same framework. In summary, this framework provides a more robust neural network-based approach than the NDM introduced in Chapter 7 because the new framework does not depend solely on sequence-to-sequence learning. Instead the framework explicitly models the hidden dialogue intentions underlying the user's utterances and allows the agent to directly learn a dialogue policy via interaction. The need for additional supervision indicates that the NVI framework still has difficulty converging, which requires more study before it can fully succeed.

Chapter 9

Conclusions

This thesis has shown that with a right architecture design, an RNN language model can generate high quality dialogue responses by learning from human-authored examples. Unlike previous approaches that rely on constructing intermediate representations, with overly explicit linguistic annotations, the goal of this thesis has been to validate the need for these representations and remove any unused or redundant ones to improve the scalability and learnability of the system. This backward integration from an NLG component has proven effective in Spoken Dialogue System architectures because it not only mitigates the development load, and makes the system more scalable, but also demonstrates a competitive advantage on human-perceived quality over previous approaches.

The original contributions of this thesis include: the development of the *RNNLG* model, which combines the *H-LSTM*, *SC-LSTM*, and *ENC-DEC* models; the domain adaptation recipe for NLG models, which applies *data counterfeiting* and *discriminative training*; and learning-based end-to-end dialogue modelling, including *NDM* and its latent-variable variant *LIDM*.

The proposed RNNLG model enables training corpus-based NLG on dialogue act-sentence pairs without alignment annotations. Alternatively, these models optimise a surface realisation objective function but also learns these alignments jointly as a by-product of using either the gating or the attention mechanism. A corpus- and human-based evaluation across four dialogue domains demonstrate all RNNLG models outperform pervious approaches, while the top performance is achieved by the SC-LSTM generator, which employs a learnable gate to control the semantic elements while generating dialogue responses.

An adaptation recipe is proposed for NLG-based domain adaptations to facilitate training for RNNLGs under a limited data scenario. Data counterfeiting allows training an out-of-domain generator on a dataset that is counterfeited from the examples in existing domains by masking (delexicalisation) and replacing the slot-value specific surface forms. Discriminative training, on the other hand, exploits a small set of in-domain examples to correct inadequate and non-fluent realisations. Experiments based on adaptation across four domains have shown improved performance compared to baseline-based approaches. This result is also confirmed via a human evaluation.

NDM further integrates the dialogue components (backward) to the dialogue manager (or content planning) and can directly learn its conversational strategies from a human-human corpus via supervised learning. By observing the fact that deterministic models, such as NDM, cannot capture natural variation in human language effectively, we discover that LIDM can carry out a more natural conversation as a generative model. Theoretically, LIDM is well-founded on conditional VAE and Neural Variational Inference, which demonstrates a strong attractiveness due to its ability to bridge different learning paradigms and resolve the credit assignment problem. In practise, LIDM and NDM both break the custom of modelling goal-oriented dialogue, as the pipelines of independent modules communicating via dialogue act taxonomies. In fact, the experimental results showed that removing the hard-coded dialogue act taxonomies reduces the annotation load but doesn't jeopardise the learning of rational system behaviour. Both a corpus-based evaluation and a live user trial show that both NDM and LIDM can engage in natural conversations to help users accomplish tasks.

9.1 Limitations and Future Work

Arguably the greatest bottleneck for statistical approaches to dialogue system development is the collection of appropriate training data. This is especially true for goal-oriented dialogue systems, which requires in-domain data for optimal performance. For example, technical support for Apple computers may differ completely from that for Windows, due to the many differences in software and hardware. Therefore, the ease of collecting these datasets plays a crucial role in the system development process. Since RNNLG relies on annotating corpora, based on the dialogue act taxonomies, it requires experts to create the labels, which is not possible to do when crowd-sourcing the work completely (Henderson, 2015a). This also applies to several system modules such as SLU and dialogue manager. On the other hand, the proposed Wizard-of-Oz data collection enables us to collect dialogue corpora with coarse-grained annotations to train NDM and LIDM. Because the WoZ data collection procedure does not require expert knowledge, it is much easier to run through crowdsourcing platforms.

Domain scalability of statistical approaches applied to dialogue systems is still an unanswered question because the current state-of-the-art methods still focus on improvements in small domains. Although we have shown that RNNLG can be readily scaled to similar domains by the proposed adaptation recipe, it is not clear whether adaptation is still feasible if an extremely different domain is involved. In addition, the situation becomes even more difficult if the generator needs to produce a single description that covers several domains. Arguably the proposed NDM and LIDM have a chance to scale better than the pipelined systems because of their much more flexible learning ability. As a result, more work should be done to train NDM or LIDM on more complex domains to validate this proposition. A hierarchical attention mechanism would be promising in this direction.

One limitation of sequential NLG models is the inability to generate long and coherent arguments. Whether it is a long sentence encompassing several subjects, or multiple sentences that each has its own idea, most sequential NLG models cannot consistently produce complex nested sentence structures. This is due to the notorious vanishing gradient problem in RNN so that it learns to be biased toward short sentences but forgets what has been said previously. Therefore, to generate long and coherent texts, better content planning and sentence planning techniques for neural networks need to be investigated. This may involve a hierarchical process with a sequence of intent variables that are sampled from a semantic-level LSTM before sentence construction, or a structured generation model in which a latent grammar is used to produce the outputs implicitly (Yogatama et al., 2016).

The RNNLG, NDM, and LIDM all rely on the process of delexicalisation (Henderson et al., 2014c; Mrkšić et al., 2015) to replace domain-specific content words or phrases with placeholders so that the model doesn't need to learn to fill in these content words during training - It is a powerful technique which allows a massive parameter sharing between homogeneous slots and values in the ontology. However, delexicalisation is a rather crude process because it usually depends on exact string matching. This restriction prevents the model from generating complex linguistic phenomena, such as referential expressions or value-based comparisons. This simple string matching technique could also cause problems[1] when the domain becomes large and therefore limit the scalability of the system. One possible solution is to apply the pointer network (Vinyals et al., 2015) with soft string matching so that the model can learn to softly match domain-specific words and copy them into sentences. The problem with softly matching different surface forms of the same semantic is that surface forms can potentially be addressed by specialised word vectors which would be lacking in this case (Faruqui et al., 2015; Mrkšić et al., 2016a).

For NDM and LIDM, one interesting direction would be to see how much we can gain from learning via direct interactions with real users by using reinforcement learning. Although a simple corpus-based RL setting has been studied for LIDM in this thesis, RL

---

[1] For example, in a domain which contains "Chinese" as a food type and "Chinese" as an ethnic group the delexicalisation can be confusing.

in the real-world is a completely different story and there are many foreseeable difficulties: firstly, how to define the rewards itself is problematic because each reward is typically implicit (Weston, 2016) and requires a model to infer (Su et al., 2015); secondly, knowing when to learn and when not to is also difficult. For example, as system developers, we may want to prevent the model from picking up undesired behaviours such swearing or impolite responses; finally, whether the model can learn efficiently in a real-world setting is also a question. There is a lot of work on sample-efficient RL approaches recently (Su et al., 2017).

The discrete latent variable employed in LIDM can model categorical dialogue intentions. However, the results may still be sub-optimal and insufficient in capturing real-world, complex conversations since there are many intentions with several degrees of similarity. Therefore, a structured representation of latent variables could be beneficial. For example, one possible configuration of a structured latent variable might consist of a few categorical distributions and a set of Bernoulli distributions to arbitrarily mimic the dialogue act taxonomies. This structured latent variable could also be self-supervised via the snapshot learning technique introduced in this thesis so that the learning can still be done in a semi-supervised fashion to guarantee the robustness of the model.

Another limitation of NDM and LIDM, or end-to-end dialogue modelling in general, is how to ensure that the model is robust to speech recognition errors. End-to-End dialogue systems typically learn dialogue strategies concurrently with language from a collection of text corpora that is usually noise-free. This is problematic if the system is operating via a speech interface with background noise that makes recognition errors inevitable. The model could behave greedily as if it is operating in a noise-free environment even in the situation where it is not quite sure what the user is talking about. One possible solution to this problem is to try and expose the wizards to noisy speech conditions during WoZ data collections so that the model can also learn from humans, e.g., confirmation behaviour.

Despite the current limitations, the proposed RNNLG, NDM, and LIDM may be particularly useful for system designers looking to build dialogue agents with extensive capabilities. The three methods were proposed with a complementary data collection methodology in mind to reduce the cost of data collection and annotation. In addition, they are all based on a flexible, neural-network-based framework to allow rapid learning in new domains. An open problem is the extent to which extensible models like these can allow for dialogue agents to continuously improve capabilities and acquire knowledge by learning either from human-human conversations or via direct interactions.

Appendix A

Slot-based Dialogue Domains

Goal-oriented dialogue systems typically use *slot-filling* mechanism to proceed a dialogue. The slots of a slot-based dialogue system specify the domain the system can talk about and the tasks it can help users to accomplish. The slots also define the set of actions the system can take, the set of possible semantics of the user utterances, and the possible dialogue states. Most of the slots come with values that come from a set of slot-specific attributes in a domain.

In the information-seeking dialogue scenario, where the goal is to help users to search a database for data items by specifying constraints. The slots are the attributes of the items in the database and a set of slot-value pairs form a search query.

There are typically two types of slots that constitute the set of all slots $\mathrm{S} = \mathrm{S}_{\mathrm{inf}} \cup \mathrm{S}_{\mathrm{req}}$, where the $\mathrm{S}_{\mathrm{inf}}$ is the set of informable slots while $\mathrm{S}_{\mathrm{req}}$ is the set of requestable slots. Informable slots are attributes of the entities in the database that users can use to constrain a search. On the other hand, requestable slots are attributes that allows users to ask for values, but may not be used as part of the search constraint. A common example of a requestable slot is the address of an entity, which may not be specified as a search constraint ("*I'm looking for a restaurant that is located at 29 North Street.*") but the user may ask the value for (*Can I have the address of the restaurant you just mentioned?*). In addition, these two sets of slots are not necessarily disjoint. Requestable slots are typically not informable, while informable slots are typically requestable.

The four domains used for evaluating the proposed approaches in this thesis are *restaurant*, *hotel*, *laptop*, and *TV* domains. The restaurant and hotel domains were collected and experimented on Wen et al. (2015c), which is about finding a venue (either a restaurant or a hotel) in the San Francisco area. The laptop and TV domains were proposed and evaluated in Wen et al. (2015b) and they are concerned with finding a product (either a laptop or a TV) to buy. A summary of the four domains is given in Table A.1.

	Slot	$\|\|V_s\|\|$			
		Restaurant	Hotel	Laptop	TV
S_{inf}	price range	3	4	3	3
	area	155	155	-	-
	near	39	28	-	-
	food	59	-	-	-
	good for meal	4	-	-	-
	kids allowed	2	-	-	-
	has internet	-	2	-	-
	accept cards	-	2	-	-
	dogs allowed	-	2	-	-
	family	-	-	4	12
	battery rating	-	-	3	-
	drive range	-	-	3	-
	weight range	-	-	3	-
	is for business	-	-	2	-
	screen size range	-	-	-	3
	eco rating	-	-	-	5
	hdmi port	-	-	-	4
	has usb port	-	-	-	2
$S_{req} \backslash S_{inf}$	type	1	1	1	1
	name	239	180	123	94
	price	96	-	79	11
	address	239	180	-	-
	phone	239	180	-	-
	postcode	21	19	-	-
	warranty	-	-	3	-
	battery	-	-	16	-
	design	-	-	20	-
	dimension	-	-	19	-
	utility	-	-	7	-
	weight	-	-	24	-
	platform	-	-	5	-
	memory	-	-	7	-
	drive	-	-	7	-
	processor	-	-	10	-
	resolution	-	-	-	3
	power consumption	-	-	-	24
	accessories	-	-	-	7
	color	-	-	-	22
	screen size	-	-	-	12
	audio	-	-	-	3

Table A.1 Slots in the restaurant, hotel, laptop, and TV domains respectively. All the informable slots are also requestable. The group of $S_{req} \backslash S_{inf}$ shows the requestable slots that are not informable. Note the type slot contains only one value, which is the domain string.

Appendix B

Dialogue Act Format

A dialogue act can be considered a shallow representation of the semantics of either a user's utterance or a system's prompt. On the input side, the dialogue system understands natural language by mapping text into one of the dialogue act taxonomies to represent the input semantics. The output side dialogue acts represent system actions or intentions that are latter transcribed back to text via the Natural Language Generation component. Therefore, both the input and output semantics are constrained with a particular scope for facilitating rational system behaviours within a domain. Multiple formats have been proposed and employed 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, goal-oriented dialogues.

A dialogue act consists of two components: a dialogue act type *da*, and a set of slot-value pairs. As mentioned in Appendix A, the set of all possible slots within a domain is the union of two sets $\mathrm{S} = \mathrm{S}_{\mathrm{inf}} \cup \mathrm{S}_{\mathrm{req}}$, the set of all informable slots $\mathrm{S}_{\mathrm{inf}}$ and the set of all requestable slots $\mathrm{S}_{\mathrm{req}}$. In a dialogue act, each slot $\mathrm{s} \in \mathrm{S}$ can either bind to a value "$\mathrm{s} = \mathrm{v}$", bind to a particular value but negated "$\mathrm{s} \neq \mathrm{v}$" (**bounded slots**), or simply doesn't bind to any value and is written as "s" (**unbounded slots**), where $\mathrm{v} \in |\mathrm{V}_{\mathrm{s}}|$ and V_{s} denote the set of all possible values for slot s.

Consider a simple example of *da = request*. The set of slot-value pairs is *{name = seven days, food = chinese, price range != expensive, address, postcode}*. This dialogue act can be written in shorthand notation as follows: "*request(address, postcode, name = seven days, food = chinese, price range != expensive)*". It represents the abstract meaning of the following description "*Can you tell me the address and postcode of the inexpensive chinese restaurant called Seven Days?* " The two unbounded slots are *address* and *postcode* and the other slots are all bounded by a maximal value. Different dialogue act types impose restrictions on what must be contained in its slot-value pairs. They are used differently, depending on whether they are a user request or a system prompt. This thesis focuses on the Natural Language Generation component of the system therefore we emphasis on dialogue acts for system prompts, as shown in Table B.1. Note that Table B.1 is grouped by the restrictions imposed on the corresponding slot-value pairs.

Restrictions on s-v pairs Z	**Act Type**	**Description**
Z must be empty	hello bye reqmore	A welcome message to start the dialogue. A goodbye message to end the dialogue. Asking whether more information is required.
$Z = \{s = v, ...\}$ only contains bounded slots.	inform inform_all inform_no_info inform_no_match confirm	Informing the requested information of a given entity as specified in Z. Informing that all the entities with specific constraints fulfils a requirement. Informing that the system cannot help with that specific information. Informing that there are *no* entities that match the constraint. confirming that the user wants a venue matching the bound slot constraints.
Z contain one *name* slot plus a set of bounded slots	recommend inform_only_match	Offering or recommending an entity and informing its attributes as provided in Z. Informing that the only match is a particular entity.
Z contain one *count* slot plus a set of bounded slots	inform_count	Informing that there are *count* entities that match the constraint.
$Z = \{s, ...\}$, Z may contain only unbounded slots	request	Requesting what the user wants for unbounded slots.
$Z = \{s = v_0, s = v_1, ...\}$, Z may contain several pairs of the same slot with different values	select suggest	Asking the user to select between the two suggested values for the slot. Asking the user to select between the three suggested values for the slot.
$Z = \{n = n_0, n = n_1, s = v_0, s = v_1, ...\}$, Z contains two name slots and two set of bounded slots	compare	Comparing two entities with the same set of attributes.

Table B.1 The set of dialogue acts used in RNNLG. Note that the dialogue act definition here is different from Young (2007) in which the *inform* act is broken down into specific categories to facilitate data collection and learning.

Appendix C

Example of a Template-based Generator

The template-based generator introduced here is based on a recursive, top-down realisation strategy. Each rule is a mapping from the concept level string to a final surface form which contains only words, or an intermediate sentence structure that consists of both vocabulary words and lower level concept strings. Based on the input dialogue act, the generator first chooses the root rule to apply by matching the input dialogue act against a set of pre-defined rules. Once a matching root rule is found, it is directly applied to map the dialogue act into the target string. If the mapped string contains only words, it is treated directly as the final surface form and subsequently passed to the speech synthesis component. However, if lower level concept strings are found, the generator needs to apply the rules to each string recursively until no abstract concept string is left. Table C.1 and Table C.2 show a few snapshots of the template-based generator used in the restaurant domain.

Table C.1 shows a set of root rules for mapping the input dialogue act into a target sentence structure. Both the slot and value could be variables (annotated by the $ sign). These variables can be directly realised by replacing the $ sign variable in the sentence with the corresponding value specified in the dialogue act.

Whenever a % sign is encountered, the model needs to look for another rule specified in the rule set to recursively map the result back to natural language. This can then make use of the lower-level concept rules shown in Table C.2. Depending on the value the input dialogue act holds, the concept can be mapped to different realisations.

Root Rules	
request(area)	What part of town do you have in mind?
request(name)	What is the name of the place you are looking for?
request(pricerange)	Would you like something in the cheap, moderate, or expensive price range?
request(food)	What kind of food would you like?
...	...
request(area, option=$X)	What part of town do you have in mind? For example, something in the area of $X?
request(name, option=$X)	What is the name of the place you are looking for? For example, $X?
request(food, option=$X)	What kind of food would you like? For example, $X?
...	...
inform(name=$X,$Y=$O)	$X is a nice place %$Y_inf($O).
inform(name=$X,$Y=$O,$Z=$P)	$X is a nice place, %$Y_inf($O) and %$Z_inf($P).
inform(name=$X,$Y=$O,$Z=$P,$W=$R)	$X is a nice place, %$Y_inf($O), %$Z_inf($P) and %$W_inf($R).
...	...
select(food=$X, food=dontcare)	Sorry would you like $X food or you don't care?
select(food=$X, food=$Y)	Sorry would you like $X or $Y food?
select($X=$Y,$X=dontcare)	Sorry would you like $Y for $X or you don't care?
select($X=$Y,$X=$Z)	Sorry would you like $Y or $Z for $X?
...	...
...	...

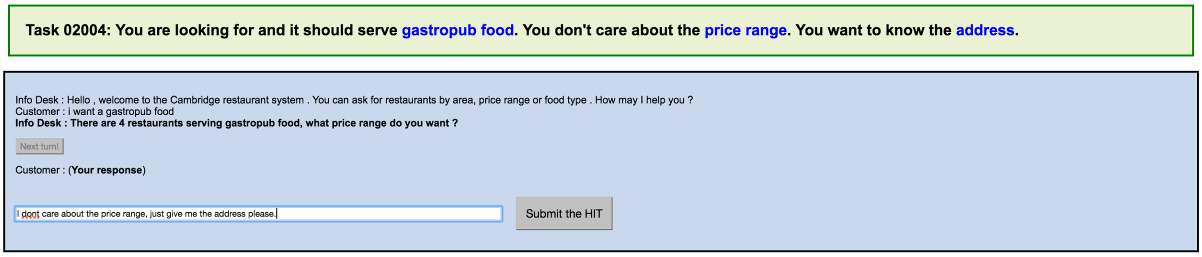
Table C.1 A sample snapshot of the root rules of the template-based generator used in this thesis. Note that both slot and value can be variables. The matching is done by matching as many slot-value pairs as possible.

Lower Level Concept Rules		
Concept String	Arguments	Realisation
area_inf($area)	$area=centre	it is in the centre of town
	$area=north	it is in the north part of town
	$area=south	it is in the south part of town
	...	...
	$area=none	I don’t know the area
	$area=$area	it is in the area of $area
dogsallowed_str($allow)	$allow=dontcare	if you don’t care whether dogs are allowed
	$allow=1	where dogs are allowed
	$allow=0	where no dogs are allowed
	$allow=none	I don’t know if it allows dogs
phone_str($phone)	$phone=none	I don’t know their address
	$phone=$phone	Their address is $addr
address_str($address)	$address=none	I don’t know their phone number
	$address=$addr	Their phone number is $phone
postcode_str($code)	$code=none	I don’t know their post code
	$code=$code	Their postcode is $code
...	...	...

Table C.2 A sample snapshot of the lower level rules.

Appendix D

The Wizard-of-Oz website



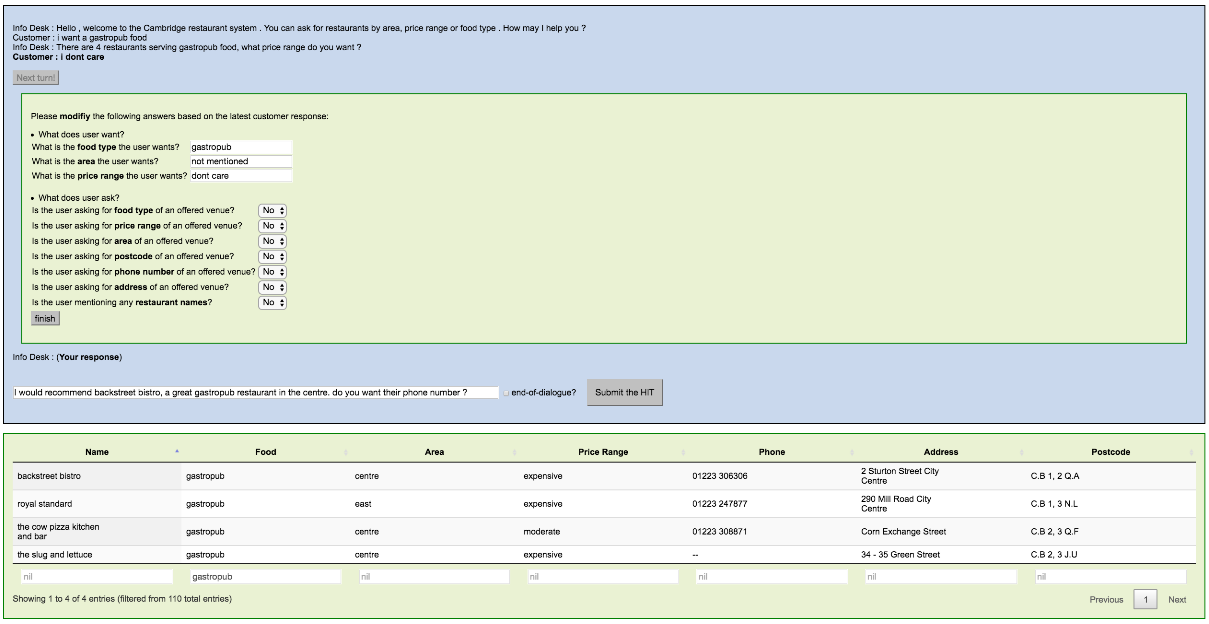
Task 02004: You are looking for and it should serve gastropub food. You don't care about the price range. You want to know the address.

Info Desk : Hello , welcome to the Cambridge restaurant system . You can ask for restaurants by area, price range or food type . How may I help you ?
Customer : i want a gastropub food
Info Desk : There are 4 restaurants serving gastropub food, what price range do you want ?

Customer : (**Your response**)

I dont care about the price range, just give me the address please.

Fig. D.1 The user webpage. The worker who plays a user is given a task to follow. For each mturk HIT, he/she needs to type an appropriate sentence to carry on the dialogue by looking at both the task description and the dialogue history.



Info Desk : Hello , welcome to the Cambridge restaurant system . You can ask for restaurants by area, price range or food type . How may I help you ?
Customer : i want a gastropub food
Info Desk : There are 4 restaurants serving gastropub food, what price range do you want ?
Customer : i dont care

Please **modifiy** the following answers based on the latest customer response:

- What does user want?

What is the **food type** the user wants? gastropub
What is the **area** the user wants? not mentioned
What is the **price range** the user wants? dont care

- What does user ask?

Is the user asking for **food type** of an offered venue? No
Is the user asking for **price range** of an offered venue? No
Is the user asking for **area** of an offered venue? No
Is the user asking for **postcode** of an offered venue? No
Is the user asking for **phone number** of an offered venue? No
Is the user asking for **address** of an offered venue? No
Is the user mentioning any **restaurant names**? No

Info Desk : (**Your response**)

I would recommend backstreet bistro, a great gastropub restaurant in the centre. do you want their phone number ?

☐ end-of-dialogue?

Name	Food	Area	Price Range	Phone	Address	Postcode
backstreet bistro	gastropub	centre	expensive	01223 306306	2 Sturton Street City Centre	C.B 1, 2 Q.A
royal standard	gastropub	east	expensive	01223 247877	290 Mill Road City Centre	C.B 1, 3 N.L
the cow pizza kitchen and bar	gastropub	centre	moderate	01223 308871	Corn Exchange Street	C.B 2, 3 Q.F
the slug and lettuce	gastropub	centre	expensive	--	34 - 35 Green Street	C.B 2, 3 J.U
nil	gastropub	nil	nil	nil	nil	nil

Showing 1 to 4 of 4 entries (filtered from 110 total entries)

Fig. D.2 The wizard page. The wizard's job is slightly more complex: the worker needs to go through the dialogue history, fill in the form (top green) by interpreting the user input at this turn, and type in an appropriate response based on the history and the DB result (bottom green). The DB search result is updated when the form is submitted. The form can be divided into informable slots (top) and requestable slots (bottom), which contain all the labels we need to train the trackers.

Acronyms

AMT Amazon Mechanical Turk. 3, 50, 51, 54, 57

ANN Artificial Neural Network. 27

ASR Automatic Speech Recognition. 6, 7, 9, 10

BPTT Back-propagation through time. 30, 34, 35

CCG Combinatorial Categorical Grammar. 7

CNN Convolutional Neural Network. xiv, 7–9, 35–37, 39

CV Computer Vision. 62

DA dialogue act. xiii, xiv, 7, 8, 12, 14, 18, 19, 21, 22, 24, 25, 43–49, 51–54, 57, 59, 60

DBN Dynamic Bayesian Network. xiii, 8, 9, 19, 20, 25, 53

DNN Deep Neural Network. 62

DST Dialogue State Tracking. 9, 10

DSTC Dialogue State Tracking Challenge. 9, 10

ENC-DEC Attention-based Encoder Decoder. xiv, 3, 48, 49, 53, 56, 58, 59

FNN Feed-forward Neural Network. xiv, 27, 30, 36, 37

GP Gaussian Process. 11

H-LSTM Heuristically Gated LSTM. xiv, 3, 45, 46, 56–60

HMM Hidden Markov Model. 6

kNN k-Nearest Neighbour. 21, 52

LIDM Latent Intention Dialogue Model. 4

LSTM Long Short-term Memory. xiv, 32–34, 45–50, 53, 57, 59

MDP Markov Decision Process. 10

MLE Maximum Likelihood Estimation. 37

MLP Multi-layer Perceptron. 38, 39

MT Machine Translation. 22, 23, 32, 35, 48

NDM Neural Dialogue Model. 4

NLG Natural Language Generation. xiii, 2–4, 11–17, 19–25, 52, 61, 62, 64, 66, 74, 77, 78

NLP Natural Language Processing. 48, 62

NN Neural Network. 38, 63

NVI Neural Variational Inference. 38, 39

POMDP Partially Observable Markov Decision Process. 4, 10

RL Reinforcement Learning. 11

RNN Recurrent Neural Network. xiii, 2, 3, 6, 8, 9, 25, 30–35, 38, 39, 43, 44, 48, 64, 74

RNNLG Recurrent Neural Network Language Generation. xiv, xviii, 3, 43, 44, 52–54, 57–59, 66, 74, 79

RST Rhetorical Structure Theory. 14

SC-LSTM Semantically Conditioned LSTM. xiv, 3, 4, 45, 47, 48, 56–60, 68, 73, 74

SDS Spoken Dialogue System. 3, 5, 6, 11

SGD Stochastic Gradient Decent. 29

SLU Spoken Language Understanding. 7–10

SVD Singular Value Decomposition. 63

SVM Support Vector Machine. 21, 63

VAE Variational Autoencoder. 4, 38, 39

WoZ Wizard-of-Oz. xi, 85, 86

References

Allen, J. (1995). *Natural language understanding*. Pearson.

Amodei, D., Anubhai, R., Battenberg, E., Case, C., Casper, J., Catanzaro, B., Chen, J., Chrzanowski, M., Coates, A., Diamos, G., Elsen, E., Engel, J., Fan, L., Fougner, C., Han, T., Hannun, A. Y., Jun, B., LeGresley, P., Lin, L., Narang, S., Ng, A. Y., Ozair, S., Prenger, R., Raiman, J., Satheesh, S., Seetapun, D., Sengupta, S., Wang, Y., Wang, Z., Wang, C., Xiao, B., Yogatama, D., Zhan, J., and Zhu, Z. (2016). Deep speech 2: End-to-end speech recognition in english and mandarin. In *ICML*, ICML'16, pages 173–182. JMLR.org.

Angeli, G., Liang, P., and Klein, D. (2010). A simple domain-independent probabilistic approach to generation. In *EMNLP*, pages 502–512, Cambridge, MA. ACL.

Auli, M. and Gao, J. (2014). Decoder integration and expected bleu training for recurrent neural network language models. In *Proceedings of the 52nd Annual Meeting of the ACL (Volume 2: Short Papers)*, pages 136–142, Baltimore, Maryland. ACL.

Bahdanau, D., Cho, K., and Bengio, Y. (2015). Neural machine translation by jointly learning to align and translate. *ICLR*.

Baldi, P., Brunak, S., Frasconi, P., Pollastri, G., and Soda, G. (1999). Exploiting the past and the future in protein secondary structure prediction. *Bioinformatics*, 15:937–946.

Bastien, F., Lamblin, P., Pascanu, R., Bergstra, J., Goodfellow, I. J., Bergeron, A., Bouchard, N., and Bengio, Y. (2012). Theano: new features and speed improvements. Deep Learning and Unsupervised Feature Learning NIPS 2012 Workshop.

Belz, A. (2006). Comparing automatic and human evaluation of nlg systems. In *In Proc. EACL06*, pages 313–320.

Belz, A. (2008). Automatic generation of weather forecast texts using comprehensive probabilistic generation-space models. *Natural Language Engineering*.

Bengio, Y., Ducharme, R., Vincent, P., and Janvin, C. (2003). A neural probabilistic language model. *Journal of Machine Learning Research*.

Bengio, Y., Simard, P., and Frasconi, P. (1994). Learning long-term dependencies with gradient descent is difficult. *Neural Networks, IEEE Transactions on*.

Berant, J., Chou, A., Frostig, R., and Liang, P. (2013). Semantic parsing on freebase from question-answer pairs. In *EMNLP*.

Bergstra, J., Breuleux, O., Bastien, F., Lamblin, P., Pascanu, R., Desjardins, G., Turian, J., Warde-Farley, D., and Bengio, Y. (2010). Theano: a CPU and GPU math expression compiler. In *Proceedings of the Python for Scientific Computing Conference (SciPy)*. Oral Presentation.

Blitzer, J., McDonald, R., and Pereira, F. (2006). Domain adaptation with structural correspondence learning. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, EMNLP '06, pages 120–128, Stroudsburg, PA, USA. Association for Computational Linguistics.

Bohus, D. and Rudnicky, A. I. (1999). *Recent Trends in Discourse and Dialogue*, chapter Sorry, I didnt catch that! Springer.

Bordes, A., Usunier, N., Chopra, S., and Weston, J. (2015). Large-scale simple question answering with memory networks. *CoRR*, abs/1506.02075.

Bordes, A. and Weston, J. (2017). Learning end-to-end goal-oriented dialog. In *ICLR*.

Bunt, H., Alexandersson, J., Carletta, J., Choe, J.-W., Chengyu Fang, A., Hasida, K., Lee, K., Petukhova, V., Popescu-Belis, A., Romary, L., Soria, C., and Traum, D. (2010). Towards an ISO Standard for Dialogue Act Annotation. In *Seventh conference on International Language Resources and Evaluation (LREC'10)*, La Valette, Malta.

Busemann, S. and Horacek, H. (1998). A flexible shallow approach to text generation. *CoRR*, cs.CL/9812018.

Cao, K. and Clark, S. (2017). Latent variable dialogue models and their diversity. In *EACL*.

Celikyilmaz, A. and Hakkani-Tur, D. (2015). Convolutional neural network based semantic tagging with entity embeddings. In *NIPS Workshop on Machine Learning for SLU and Interaction*.

Chambers, N., Allen, J., HUMAN, F. I. F., and FL., M. C. I. P. (2004). *Stochastic Language Generation in a Dialogue System: Toward a Domain Independent Generator*. Defense Technical Information Center.

Chatfield, K., Simonyan, K., Vedaldi, A., and Zisserman, A. (2014). Return of the devil in the details: Delving deep into convolutional nets. In *British Machine Vision Conference*.

Chen, J. and Chaudhari, N. S. (2004). Capturing long-term dependencies for protein secondary structure prediction. In *Advances in Neural Networks - ISNN 2004, International Symposiumon Neural Networks, Part II, volume 3174 of Lecture Notes in Computer Science*, pages 494–500. Springer.

Chen, Y., Hakkani-Tür, D. Z., and He, X. (2016). Zero-shot learning of intent embeddings for expansion by convolutional deep structured semantic models. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2016*, pages 6045–6049.

Cheyer, A. and Guzzoni, D. (2007). Method and apparatus for building an intelligent automated assistant. US Patent App. 11/518,292.

Chiang, D., Andreas, J., Bauer, D., Hermann, K. M., Jones, B., and Knight, K. (2013). Parsing graphs with hyperedge replacement grammars. In *In Proceedings of the 51st Meeting of the ACL. Marie-Catherine de Marneffe*.

Cho, K., van Merrienboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. (2014). Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *EMNLP*, pages 1724–1734, Doha, Qatar. ACL.

Chung, J., Gülçehre, Ç., Cho, K., and Bengio, Y. (2014). Empirical evaluation of gated recurrent neural networks on sequence modeling. *CoRR*, abs/1412.3555.

Ciresan, D., Meier, U., Gambardella, L., and Schmidhuber, J. (2011). Convolutional neural network committees for handwritten character classification. In *Document Analysis and Recognition (ICDAR), 2011 International Conference on*, pages 1135–1139.

Collins, M. (2002). Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proceedings of the ACL-02 Conference on Empirical Methods in Natural Language Processing - Volume 10*, EMNLP '02, pages 1–8, Stroudsburg, PA, USA. Association for Computational Linguistics.

Collobert, R. and Weston, J. (2008). A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, pages 160–167, New York, NY, USA. ACM.

Craven, M. and Kumlien, J. (1999). Constructing biological knowledge bases by extracting information from text sources. In *ISMB*.

Cuayahuitl, H., Dethlefs, N., Hastie, H., and Liu, X. (2014). Training a statistical surface realiser from automatic slot labelling. In *2014 IEEE Spoken Language Technology Workshop (SLT)*, pages 112–117.

Daume III, H. (2007). Frustratingly easy domain adaptation. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 256–263, Prague, Czech Republic. Association for Computational Linguistics.

Deng, L., Li, J., Huang, J.-T., Yao, K., Yu, D., Seide, F., Seltzer, M., Zweig, G., He, X., Williams, J., Gong, Y., Acero, A., and Seltzer, M. (2013). Recent advances in deep learning for speech research at microsoft. In *ICASSP*.

der Maaten, L. V. and Hinton, G. (2008). Visualizing Data using t-SNE. *JMLR*.

Dethlefs, N., Hastie, H., Cuayahuitl, H., and Lemon, O. (2013). Conditional random fields for responsive surface realisation using global features. In *ACL*.

Doddington, G. (2002). Automatic evaluation of machine translation quality using n-gram co-occurrence statistics. In *Proceedings of the Second International Conference on Human Language Technology Research*, HLT '02, pages 138–145, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Doersch, C. (2016). Tutorial on Variational Autoencoders. *ArXiv e-prints*.

Dušek, O. and Jurcicek, F. (2016). Sequence-to-sequence generation for spoken dialogue via deep syntax trees and strings. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 45–51. Association for Computational Linguistics.

Dušek, O. and Jurcicek, F. (2015). Training a natural language generator from unaligned data. In *ACL*, pages 451–461, Beijing, China.

Dušek, O. and Jurcicek, F. (2016). Sequence-to-sequence generation for spoken dialogue via deep syntax trees and strings. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 45–51, Berlin, Germany. Association for Computational Linguistics.

Elman, J. L. (1990). Finding structure in time. *COGNITIVE SCIENCE*, 14(2):179–211.

Epstein, R. (1992). The quest for the thinking computer. *AI Mag.*, 13(2):81–95.

Espinosa, D., White, M., and Mehay, D. (2008). Hypertagging: Supertagging for surface realization with ccg. In *In Proceedings of the 40th Annual Meeting of the Association of Computational Linguistics (ACL): Human Language Technologies (HLT*, pages 183–191.

Faruqui, M., Dodge, J., Jauhar, S. K., Dyer, C., Hovy, E., and Smith, N. A. (2015). Retrofitting word vectors to semantic lexicons. In *NAACL-HLT*, pages 1606–1615, Denver, Colorado. Association for Computational Linguistics.

Frühwirth-Schnatter, S. (1994). Data augmentation and dynamic linear models. *Journal of time series analysis*, 15(2):183–202.

Gales, M. and Woodland, P. (1996). Mean and variance adaptation within the mllr framework. *Computer Speech and Language*, 10(4):249 – 264.

Gales, M. and Young, S. (2007). The application of hidden markov models in speech recognition. *Found. Trends Signal Process.*, 1(3):195–304.

Gašić, M., Breslin, C., Henderson, M., Kim, D., Szummer, M., Thomson, B., Tsiakoulis, P., and Young, S. (2013). On-line policy optimisation of bayesian spoken dialogue systems via human interaction. In *ICASSP*.

Gašić, M., Mrkšić, N., Su, P.-H., Vandyke, D., Wen, T.-H., and Young, S. J. (2015). Policy committee for adaptation in multi-domain spoken dialogue systems. In *ASRU*.

Gasic, M. and Young, S. (2014). Gaussian processes for pomdp-based dialogue manager optimization. *IEEE/ACM Transaction on Audio, Speech and Language*.

Gauvain, J.-L. and Lee, C.-H. (1994). Maximum a posteriori estimation for multivariate gaussian mixture observations of markov chains. *IEEE transactions on speech and audio processing*, 2(2):291–298.

Glynn, P. W. (1990). Likelihood ratio gradient estimation for stochastic systems. *Commun. ACM*, 33(10):75–84.

Goddeau, D., Meng, H., Polifroni, J., Seneff, S., and Busayapongchai, S. (1996). A form-based dialogue manager for spoken language applications. In *Spoken Language, 1996. ICSLP 96. Proceedings., Fourth International Conference on*, volume 2, pages 701–704 vol.2.

Goodman, J. T. (2001). A bit of progress in language modeling. *Computer Speech Language*, 15(4):403–434.

Graves, A. and Jaitly, N. (2014). Towards end-to-end speech recognition with recurrent neural networks. In Jebara, T. and Xing, E. P., editors, *ICML*, pages 1764–1772. JMLR Workshop and Conference Proceedings.

Graves, A., Liwicki, M., Fernández, S., Bertolami, R., Bunke, H., and Schmidhuber, J. (2009). A novel connectionist system for unconstrained handwriting recognition. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*.

Graves, A., Mohamed, A., and Hinton, G. E. (2013a). Speech recognition with deep recurrent neural networks. *CoRR*, abs/1303.5778.

Graves, A., Mohamed, A.-r., and Hinton, G. (2013b). Speech recognition with deep recurrent neural networks. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*.

Graves, A., Wayne, G., and Danihelka, I. (2014). Neural turing machines. *CoRR*, abs/1410.5401.

Hammer, B. (1998). On the approximation capability of recurrent neural networks. In *In International Symposium on Neural Computation*, pages 12–4.

He, X. and Deng, L. (2012). Maximum expected bleu training of phrase and lexicon translation models. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers - Volume 1*, ACL '12, pages 292–301, Stroudsburg, PA, USA. Association for Computational Linguistics.

He, Y. and Young, S. (2006). Spoken language understanding using the hidden vector state model. *Speech Communication*, 48(3-4):262–275.

Heidel, A. and Lee, L.-s. (2007). Robust topic inference for latent semantic language model adaptation. In *2007 IEEE Workshop on Automatic Speech Recognition Understanding (ASRU)*, pages 177–182.

Henderson, M. (2015a). *Discriminative Methods for Statistical Spoken Dialogue Systems*. PhD thesis, University of Cambridge.

Henderson, M. (2015b). Machine learning for dialog state tracking: A review. In *Machine Learning in Spoken Language Processing Workshop*.

Henderson, M., Thomson, B., and Williams, J. (2014a). The Second Dialog State Tracking Challenge. In *Proceedings of SIGdial*.

Henderson, M., Thomson, B., and Williams, J. (2014b). The Third Dialog State Tracking Challenge. In *Proceedings of IEEE Spoken Language Technology*.

Henderson, M., Thomson, B., and Young, S. (2013). Deep Neural Network Approach for the Dialog State Tracking Challenge. In *Proceedings of SIGdial*.

Henderson, M., Thomson, B., and Young, S. (2014c). Word-based dialog state tracking with recurrent neural networks. In *SIGdial*, pages 292–299, Philadelphia, PA, U.S.A. ACL.

Higgins, I., Matthey, L., Pal, A., Burgess, C., Glorot, X., Botvinick, M., Mohamed, S., and Lerchner, A. (2017). beta-vae: Learning basic visual concepts with a constrained variational framework. In *ICLR*.

Hill, F., Bordes, A., Chopra, S., and Weston, J. (2016). The goldilocks principle: Reading children's books with explicit memory representations. In *ICLR*.

Hinton, G., Deng, L., Yu, D., Dahl, G., rahman Mohamed, A., Jaitly, N., Senior, A., Vanhoucke, V., Nguyen, P., Sainath, T., and Kingsbury, B. (2012). Deep neural networks for acoustic modeling in speech recognition. *Signal Processing Magazine*.

Hochreiter, S., Bengio, Y., Frasconi, P., and Schmidhuber, J. (2001). Gradient flow in recurrent nets: the difficulty of learning long-term dependencies.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*.

Hoffmann, R., Zhang, C., Ling, X., Zettlemoyer, L., and Weld, D. S. (2011). Knowledge-based weak supervision for information extraction of overlapping relations. In *ACL*.

Hogan, D., Foster, J., Wagner, J., and van Genabith, J. (2008). Parser-based retraining for domain adaptation of probabilistic generators. In *INLG*, pages 165–168, Stroudsburg, PA, USA. ACL.

Hubel, D. H. and Wiesel, T. N. (1968). Receptive fields and functional architecture of monkey striate cortex. *The Journal of Physiology*, 195(1):215–243.

Jaeger, H. (2001). The "echo state" approach to analysing and training recurrent neural networks. GMD Report 148, GMD - German National Research Institute for Computer Science.

Jang, E., Gu, S., and Poole, B. (2016). Categorical reparameterization with gumbel-softmax. In *ICLR*.

Jeong, M. and Geunbae Lee, G. (2008). Triangular-chain conditional random fields. *Trans. Audio, Speech and Lang. Proc.*, 16(7):1287–1302.

Jordan, M. I. (1989). Serial order: A parallel, distributed processing approach. In *Advances in Connectionist Theory: Speech*. Lawrence Erlbaum Associates.

Jordan, M. I. (1990). Attractor dynamics and parallelism in a connectionist sequential machine. In Diederich, J., editor, *Artificial Neural Networks*, pages 112–127. IEEE Press, Piscataway, NJ, USA.

Jozefowicz, R., Zaremba, W., and Sutskever, I. (2015). An empirical exploration of recurrent network architectures. *Journal of Machine Learning Research*.

Jurčíček, F., Thomson, B., and Young, S. (2011). Natural actor and belief critic: Reinforcement algorithm for learning parameters of dialogue systems modelled as pomdps. *ACM Trans. Speech Lang. Process.*, 7(3):6:1–6:26.

Kalchbrenner, N., Grefenstette, E., and Blunsom, P. (2014). A convolutional neural network for modelling sentences. *ACL.*

Karpathy, A. and Fei-Fei, L. (2014). Deep visual-semantic alignments for generating image descriptions. *CoRR.*

Kate, R. J. and Mooney, R. J. (2006). Using string-kernels for learning semantic parsers. In *ACL 2006: Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the ACL*, pages 913–920, Morristown, NJ, USA. Association for Computational Linguistics.

Kelley, J. F. (1984). An iterative design methodology for user-friendly natural language office information applications. *ACM Transaction on Information Systems.*

Kim, S., D'Haro, L. F., Banchs, R. E., Williams, J., and Henderson, M. (2016a). The Fourth Dialog State Tracking Challenge. In *Proceedings of the 7th International Workshop on Spoken Dialogue Systems (IWSDS).*

Kim, S., D'Haro, L. F., Banchs, R. E., Williams, J., Henderson, M., and Yoshino, K. (2016b). The Fifth Dialog State Tracking Challenge. In *Proceedings of the 2016 IEEE Workshop on Spoken Language Technology (SLT).*

Kim, Y. (2014). Convolutional neural networks for sentence classification. In *EMNLP.*

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

Kingma, D. P. and Welling, M. (2014). Stochastic backpropagation and approximate inference in deep generative models. In *ICML.*

Konda, V. R. and Tsitsiklis, J. N. (2003). On actor-critic algorithms. *SIAM J. Control Optim.*, 42(4):1143–1166.

Kondadadi, R., Howald, B., and Schilder, F. (2013). A statistical nlg framework for aggregated planning and realization. In *ACL*, pages 1406–1415, Sofia, Bulgaria. ACL.

Konstas, I. and Lapata, M. (2012). Unsupervised concept-to-text generation with hypergraphs. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, NAACL HLT '12, pages 752–761, Stroudsburg, PA, USA. Association for Computational Linguistics.

Kočiský, T., Melis, G., Grefenstette, E., Dyer, C., Ling, W., Blunsom, P., and Hermann, K. M. (2016). Semantic parsing with semi-supervised sequential autoencoders. In *EMNLP*, pages 1078–1087, Austin, Texas. Association for Computational Linguistics.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In Pereira, F., Burges, C. J. C., Bottou, L., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc.

Kukich, K. (1987). Where do phrases come from: Some preliminary experiments in connectionist phrase generation. In *Natural Language Generation*. Springer Netherlands.

Kuo, H. K. J., Fosler-Lussier, E., Jiang, H., and Lee, C. H. (2002). Discriminative training of language models for speech recognition. In *2002 IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 1, pages I–325–I–328.

Lafferty, J. D., McCallum, A., and Pereira, F. C. N. (2001). Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, pages 282–289, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Langkilde, I. and Knight, K. (1998). Generation that exploits corpus-based statistical knowledge. In *ACL*, pages 704–710, Montreal, Quebec, Canada. ACL.

Larsson, S. and Traum, D. R. (2000). Information state and dialogue management in the trindi dialogue move engine toolkit. *Nat. Lang. Eng.*, 6(3-4):323–340.

Lavoie, B. and Rambow, O. (1997). A fast and portable realizer for text generation systems. In *Proceedings of the Fifth Conference on Applied Natural Language Processing*, ANLC '97, pages 265–268, Stroudsburg, PA, USA. Association for Computational Linguistics.

Lecun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998a). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*.

Lecun, Y., Bottou, L., Orr, G. B., and Muller, K.-R. (1998b). Efficient backprop.

Lee, B.-J., Lim, W., Kim, D., and Kim, K.-E. (2014). Optimizing generative dialog state tracker via cascading gradient descent. In *Proceedings of the 15th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*, pages 273–281, Philadelphia, PA, U.S.A. Association for Computational Linguistics.

Lee, S. and Eskenazi, M. (2013). Recipe for building robust spoken dialog state trackers: Dialog state tracking challenge system description. In *Proceedings of the SIGDIAL 2013 Conference*, page 414–422, Metz, France. Association for Computational Linguistics, Association for Computational Linguistics.

Leggetter, C. J. and Woodland, P. C. (1995). Maximum likelihood linear regression for speaker adaptation of continuous density hidden markov models. *Computer Speech & Language*, 9(2):171–185.

Lemon, O. (2008). Adaptive natural language generation in dialogue using reinforcement learning. In *SemDial*.

Levin, E. (1995). Chronus, the next generation. *Proc. ARPA Spoken Language Technology Workshop*, pages 269–271.

Levin, E. and Pieraccini, R. (1997). A stochastic model of computer-human interaction for learning dialogue strategies. In *In EUROSPEECH 97*, pages 1883–1886.

Li, J., Galley, M., Brockett, C., Gao, J., and Dolan, B. (2016a). A diversity-promoting objective function for neural conversation models. In *NAACL-HLT*.

Li, J., Galley, M., Brockett, C., Spithourakis, G., Gao, J., and Dolan, B. (2016b). A persona-based neural conversation model. In *ACL*, pages 994–1003, Berlin, Germany. Association for Computational Linguistics.

Li, J., Monroe, W., Ritter, A., Jurafsky, D., Galley, M., and Gao, J. (2016c). Deep reinforcement learning for dialogue generation. In *EMNLP*, pages 1192–1202, Austin, Texas. Association for Computational Linguistics.

Lin, C.-Y. and Hovy, E. (2003). Automatic evaluation of summaries using n-gram co-occurrence statistics. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1*, NAACL '03, pages 71–78, Stroudsburg, PA, USA. Association for Computational Linguistics.

Liu, C., Lowe, R., Serban, I. V., Noseworthy, M., Charlin, L., and Pineau, J. (2016). How NOT to evaluate your dialogue system: An empirical study of unsupervised evaluation metrics for dialogue response generation. *CoRR*, abs/1603.08023.

Liu, J., Cyphers, D. S., Pasupat, P., McGraw, I., and Glass, J. R. (2012). A conversational movie search system based on conditional random fields. In *INTERSPEECH*.

Logan, B. (2000). Mel frequency cepstral coefficients for music modeling. In *International Symposium on Music Information Retrieval*.

Lucas, B. (2000). Voicexml for web-based distributed conversational applications. *Commun. ACM*, 43(9):53–57.

Lyons, J. (2007). Artificial stupidity. In *ACM SIGGRAPH 2007 Computer Animation Festival*, SIGGRAPH '07, pages 27–, New York, NY, USA. ACM.

Mairesse, F., Gasic, M., Jurcicek, F., Keizer, S., Thomson, B., Yu, K., and Young, S. (2009). Spoken language understanding from unaligned data using discriminative classification models. In *2009 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 4749–4752.

Mairesse, F., Gasic, M., Jurcicek, F., Keizer, S., Thomson, B., Yu, K., and Young, S. (2010). Phrase-based statistical language generation using graphical models and active learning. In *ACL*, pages 1552–1561, Uppsala, Sweden. ACL.

Mairesse, F. and Walker, M. A. (2011). Controlling user perceptions of linguistic style: Trainable generation of personality traits. *Computer Linguistics*.

Mairesse, F. and Young, S. (2014). Stochastic language generation in dialogue using factored language models. *Computer Linguistics*.

Mangu, L., Brill, E., and Stolcke, A. (2000). Finding consensus among words: lattice-based word error minimisation. *Computer Speech and Language*, pages 373–400.

Mann, W. C. and Thompson, S. A. (1988). Rhetorical structure theory: Toward a functional theory of text organization. *Text*.

McCarthy, J., Minsky, M. L., Rochester, N., and Shannon, C. E. (1955). A Proposal for the Dartmouth Summer Research Project on Artificial Intelligence.

McCulloch, W. S. and Pitts, W. (1988). A logical calculus of the ideas immanent in nervous activity. In Anderson, J. A. and Rosenfeld, E., editors, *Neurocomputing: Foundations of Research*, pages 15–27. MIT Press, Cambridge, MA, USA.

Mei, H., Bansal, M., and Walter, M. R. (2015). What to talk about and how? selective generation using lstms with coarse-to-fine alignment. *CoRR*, abs/1509.00838.

Melčuk, I. (1988). *Dependency Syntax: Theory and Practice*. Daw Book Collectors. State University Press of New York.

Mesnil, G., Dauphin, Y., Yao, K., Bengio, Y., Deng, L., Hakkani-Tur, D., He, X., Heck, L., Tur, G., Yu, D., and Zweig, G. (2015). Using recurrent neural networks for slot filling in spoken language understanding. *Trans. Audio, Speech and Lang. Proc.*, 23(3):530–539.

Meza-Ruiz, I. V., Riedel, S., and Lemon, O. (2008). Accurate statistical spoken language understanding from limited development resources. In *2008 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 5021–5024.

Miao, Y. and Blunsom, P. (2016). Language as a latent variable: Discrete generative models for sentence compression. In *EMNLP*, pages 319–328, Austin, Texas. Association for Computational Linguistics.

Miao, Y., Yu, L., and Blunsom, P. (2016). Neural variational inference for text processing. In *ICML*.

Mikolov, T., Karafiat, M., Burget, L., Černocký, J., and Khudanpur, S. (2010). Recurrent neural network based language model. In *InterSpeech*.

Mikolov, T., Kombrink, S., Burget, L., Černocký, J., and Khudanpur, S. (2011). Extensions of recurrent neural network language model. In *ICASSP*, pages 5528–5531. IEEE.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. (2013). Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems 26*.

Mikolov, T. and Zweig, G. (2012). Context dependent recurrent neural network language model. In *IEEE SLT*.

Miller, A. H., Fisch, A., Dodge, J., Karimi, A.-H., Bordes, A., and Weston, J. (2016). Key-value memory networks for directly reading documents. In *EMNLP*.

Miller, S., Schwartz, R., Bobrow, R., and Ingria, R. (1994). Statistical language processing using hidden understanding models. In *Proceedings of the Workshop on Human Language Technology*, HLT '94, pages 278–282, Stroudsburg, PA, USA. Association for Computational Linguistics.

Miller, S., Stallard, D., Bobrow, R., and Schwartz, R. (1996). A fully statistical approach to natural language interfaces. In *Proceedings of the 34th Annual Meeting on Association for Computational Linguistics*, ACL '96, pages 55–61, Stroudsburg, PA, USA. Association for Computational Linguistics.

Minsky, M. (1961). Steps toward artificial intelligence. *Proceedings of the IRE*, 49(1):8–30.

Mintz, M., Bills, S., Snow, R., and Jurafsky, D. (2009). Distant supervision for relation extraction without labeled data. In *ACL*.

Mirkovic, D. and Cavedon, L. (2011). Dialogue management using scripts. EP Patent 1,891,625.

Misu, T. and Kawahara, T. (2007). Speech-based interactive information guidance system using question-answering technique. *2007 IEEE International Conference on Acoustics, Speech and Signal Processing - ICASSP '07*, 4:IV–145–IV–148.

Mnih, A. and Gregor, K. (2014). Neural variational inference and learning in belief networks. In *ICML*.

Mnih, V., Heess, N., Graves, A., and kavukcuoglu, k. (2014). Recurrent models of visual attention. In *NIPS*.

Mrkšić, N., Ó Séaghdha, D., Thomson, B., Gasic, M., Su, P.-H., Vandyke, D., Wen, T.-H., and Young, S. (2015). Multi-domain dialog state tracking using recurrent neural networks. In *ACL*, pages 794–799, Beijing, China. ACL.

Mrkšić, N., Ó Séaghdha, D., Wen, T.-H., Thomson, B., and Young, S. (2016a). Counterfitting word vectors. In *Proceedings of NAACL*.

Mrkšić, N., Ó Séaghdha, D., Wen, T.-H., Thomson, B., and Young, S. (2016b). Neural belief tracker: Data-driven dialogue state tracking. In *Proceedings of ACL*.

Murveit, H., Butzberger, J., Digalakis, V., and Weintraub, M. (1993). Large-vocabulary dictation using sri's decipher speech recognition system: progressive search techniques. In *1993 IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 2, pages 319–322 vol.2.

Nguyen, L., Shimazu, A., and hieu Phan, X. (2006). Semantic parsing with structured svm ensemble classification models. In *In Proceedings of the COLING/ACL 2006 Main Conference Poster Sessions*, pages 619–626.

Novikova, J., Dušek, O., Cercas Curry, A., and Rieser, V. (2017). Why we need new evaluation metrics for nlg. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2241–2252. Association for Computational Linguistics.

Oh, A. H. and Rudnicky, A. I. (2000). Stochastic language generation for spoken dialogue systems. In *NAACL Workshop on Conversational Systems*.

Papineni, K., Roukos, S., Ward, T., and Zhu, W.-J. (2002). Bleu: a method for automatic evaluation of machine translation. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Parsons, T. (1990). *Events in the Semantics of English: A Study in Subatomic Semantics*. MIT Press.

Perez, J. (2016). Dialog state tracking, a machine reading approach using a memory-enhanced neural network. *CoRR*, abs/1606.04052.

Pieraccini, R. and Huerta, J. M. (2008). Where do we go from here? Research and commercial spoken dialogue systems. In *Recent Trends in Discourse and Dialogue*, volume 39 of *Text, Speech and Language Technology*, chapter 1, pages 1–24. Springer, Dordrecht.

Raiko, T., Berglund, M., Alain, G., and Dinh, L. (2014). Techniques for learning binary stochastic feedforward neural networks. *CoRR*, abs/1406.2989.

Rapaport, W. J. (1986). Logical foundations for belief representation. *Cognitive Science*, 10(4):371–422.

Ratnaparkhi, A. (2002). Trainable approaches to surface natural language generation and their application to conversational dialog systems. *Computer Speech and Language*.

Reiter, E. and Dale, R. (1997). Building applied natural language generation systems. *Nat. Lang. Eng.*, 3(1):57–87.

Reiter, E. and Dale, R. (2000). *Building Natural Language Generation Systems*. Cambridge University Press, New York, NY, USA.

Ren, H., Xu, W., and Yan, Y. (2014). Markovian discriminative modeling for cross-domain dialog state tracking. In *2014 IEEE Spoken Language Technology Workshop (SLT)*, pages 342–347.

Rezende, D. J., Mohamed, S., and Wierstra, D. (2014). Stochastic backpropagation and approximate inference in deep generative models. In *ICML*.

Rieser, V. and Lemon, O. (2010). Natural language generation as planning under uncertainty for spoken dialogue systems. In *EMNLG*. Springer-Verlag.

Rieser, V. and Lemon, O. (2011). *Reinforcement Learning for Adaptive Dialogue Systems: A Data-driven Methodology for Dialogue Management and Natural Language Generation*. Springer Publishing Company, Incorporated.

Rieser, V., Lemon, O., and Keizer, S. (2014). Natural language generation as incremental planning under uncertainty: Adaptive information presentation for statistical dialogue systems. *IEEE/ACM Trans. Audio, Speech and Lang. Proc.*, 22(5):979–994.

Robinson, A. J. and Fallside, F. (1987). The utility driven dynamic error propagation network. Technical Report CUED/F-INFENG/TR.1, Cambridge University Engineering Department, Cambridge.

Rojas Barahona, L. M., Gasic, M., Mrkšić, N., Su, P.-H., Ultes, S., Wen, T.-H., and Young, S. (2016). Exploiting sentence and context representations in deep neural models for spoken language understanding. In *Coling*, pages 258–267, Osaka, Japan. The COLING 2016 Organizing Committee.

Roy, N., Pineau, J., and Thrun, S. (2000a). Spoken dialogue management using probabilistic reasoning. In *Proceedings of the 38th Annual Meeting on Association for Computational Linguistics*, pages 93–100. Association for Computational Linguistics.

Roy, N., Pineau, J., and Thrun, S. (2000b). Spoken dialogue management using probabilistic reasoning. In *Proceedings of the 38th Annual Meeting on Association for Computational Linguistics*, ACL '00, pages 93–100, Stroudsburg, PA, USA. Association for Computational Linguistics.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning internal representations by error propagation. In Rumelhart, D. E., McClelland, J. L., and PDP Research Group, C., editors, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol. 1*, pages 318–362. MIT Press, Cambridge, MA, USA.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1988). Learning internal representations by error propagation. In Anderson, J. A. and Rosenfeld, E., editors, *Neurocomputing: Foundations of Research*, pages 673–695. MIT Press, Cambridge, MA, USA.

Sainath, T. N., Mohamed, A.-r., Kingsbury, B., and Ramabhadran, B. (2013). Deep convolutional neural networks for lvcsr. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*.

Schapire, R. E. (1999). A brief introduction to boosting. In *Proceedings of the 16th International Joint Conference on Artificial Intelligence - Volume 2*, IJCAI'99, pages 1401–1406, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Schmidhuber, J. (1992). Learning complex, extended sequences using the principle of history compression. *Neural Comput.*, 4(2):234–242.

Schuster, M. and Paliwal, K. (1997a). Bidirectional recurrent neural networks. *Trans. Sig. Proc.*, 45(11):2673–2681.

Schuster, M. and Paliwal, K. K. (1997b). Bidirectional recurrent neural networks. *Signal Processing, IEEE Transactions on*.

Searle, J. (1969). *Speech Acts: An Essay in the Philosophy of Language*. Cam: Verschiedene Aufl. Cambridge University Press.

Seneff, S. and Polifroni, J. (2000). Dialogue management in the mercury flight reservation system. In *Proceedings of the 2000 ANLP/NAACL Workshop on Conversational Systems - Volume 3*, ANLP/NAACL-ConvSyst '00, pages 11–16, Stroudsburg, PA, USA. Association for Computational Linguistics.

Serban, I. V., Lowe, R., Charlin, L., and Pineau, J. (2015a). A survey of available corpora for building data-driven dialogue systems. *arXiv preprint:1512.05742*.

Serban, I. V., Sordoni, A., Bengio, Y., Courville, A. C., and Pineau, J. (2015b). Hierarchical neural network generative models for movie dialogues. In *AAAI*.

Serban, I. V., Sordoni, A., Lowe, R., Charlin, L., Pineau, J., Courville, A., and Bengio, Y. (2016). A hierarchical latent variable encoder-decoder model for generating dialogues. *arXiv preprint: 1605.06069*.

Shang, L., Lu, Z., and Li, H. (2015). Neural responding machine for short-text conversation. In *ACL*, pages 1577–1586, Beijing, China. ACL.

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., et al. (2016). Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489.

Singh, S., Kearns, M., Litman, D., and Walker, M. (1999). Reinforcement learning for spoken dialogue systems. In *Proceedings of the 12th International Conference on Neural Information Processing Systems*, NIPS'99, pages 956–962, Cambridge, MA, USA. MIT Press.

Smith, R. (2014). Comparative error analysis of dialog state tracking. In *Proceedings of the 15th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*, pages 300–309, Philadelphia, PA, U.S.A. Association for Computational Linguistics.

Snow, R., Jurafsky, D., and Ng, A. Y. (2004). Learning syntactic patterns for automatic hypernym discovery. In *NIPS*.

Steedman, M. (2000). *The Syntactic Process*. MIT Press, Cambridge, MA, USA.

Stent, A., Marge, M., and Singhai, M. (2005). Evaluating evaluation methods for generation in the presence of variation. In *Proceedings of the 6th International Conference on Computational Linguistics and Intelligent Text Processing*, pages 341–351, Berlin, Heidelberg. Springer-Verlag.

Stent, A. and Molina, M. (2009). Evaluating automatic extraction of rules for sentence plan construction. In *SIGdial*, pages 290–297, London, UK. ACL.

Stent, A., Prassad, R., and Walker, M. (2004). Trainable sentence planning for complex information presentations in spoken dialog systems. In *ACL*, pages 79–86, Barcelona, Spain. ACL.

Su, P., Gasic, M., Mrksic, N., Rojas-Barahona, L. M., Ultes, S., Vandyke, D., Wen, T., and Young, S. J. (2016a). Continuously learning neural dialogue management. *CoRR*, abs/1606.02689.

Su, P.-H., Budzianowski, P., Ultes, S., Gasic, M., and Young, S. (2017). Sample-efficient Actor-Critic Reinforcement Learning with Supervised Data for Dialogue Management. In *Proceedings of SIGdial*.

Su, P.-H., Gašić, M., Mrkšić, N., Rojas-Barahona, L., Ultes, S., Vandyke, D., Wen, T.-H., and Young, S. (2016b). On-line active reward learning for policy optimisation in spoken dialogue systems. In *Proceedings of ACL*.

Su, P.-H., Vandyke, D., Gasic, M., Kim, D., Mrksic, N., Wen, T.-H., and Young, S. J. (2015). Learning from real users: Rating dialogue success with neural networks for reinforcement learning in spoken dialogue systems. In *Interspeech*.

Sukhbaatar, S., Szlam, A., Weston, J., and Fergus, R. (2015). End-to-end memory networks. In *NIPS*.

Sun, K., Chen, L., Zhu, S., and Yu, K. (2014). The sjtu system for dialog state tracking challenge 2. In *Proceedings of the 15th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*, pages 318–326, Philadelphia, PA, U.S.A. Association for Computational Linguistics.

Sundermeyer, M., Alkhouli, T., Wuebker, J., and Ney, H. (2014). Translation modeling with bidirectional recurrent neural networks. In *EMNLP*. ACL.

Sutskever, I., Martens, J., and Hinton, G. (2011). Generating text with recurrent neural networks. In *ICML*, ICML '11, pages 1017–1024, New York, NY, USA. ACM.

Sutskever, I., Vinyals, O., and Le, Q. V. (2014). Sequence to sequence learning with neural networks. In *NIPS*, NIPS'14, pages 3104–3112, Cambridge, MA, USA. MIT Press.

Sutton, R. S. and Barto, A. G. (1998). *Introduction to Reinforcement Learning*. MIT Press, Cambridge, MA, USA, 1st edition.

Sutton, S., Novick, D., Cole, R., Vermeulen, P., de Villiers, J., Schalkwyk, J., and Fanty, M. (1996). *Building 10,000 spoken dialogue systems*, volume 2, pages 709–712. IEEE.

Tanner, M. A. and Wong, W. H. (1987). The calculation of posterior distributions by data augmentation. *Journal of the American statistical Association*, 82(398):528–540.

Taylor, P., Black, A. W., and Caley, R. (1998). The architecture of the festival speech synthesis system. In *The Third ESCA Workshop in Speech Synthesis*, pages 147–151.

Thomson, B. and Young, S. (2010). Bayesian update of dialogue state: A pomdp framework for spoken dialogue systems. *Comput. Speech Lang.*, 24(4):562–588.

Traum, D. R. (1999). *Foundations of Rational Agency*, chapter Speech Acts for Dialogue Agents. Springer.

Traum, D. R. (2000). 20 questions on dialogue act taxonomies. *Journal of semantics*, 17(1):7–30.

Tsiakoulis, P., Breslin, C., Gašić, M., Henderson, M., Kim, D., Szummer, M., Thomson, B., and Young, S. (2014). Dialogue context sensitive hmm-based speech synthesis. In *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2554–2558.

Tur, G. and Deoras, A. (2013). Semantic parsing using word confusion networks with conditional random fields. In *In Proc. of Interspeech*.

Turing, A. M. (1950). Computing machinery and intelligence. *Mind*, 59(236):433–460.

Van Deemter, K., Krahmer, E., and Theune, M. (2005). Real versus template-based natural language generation: A false opposition? *Comput. Linguist.*, 31(1):15–24.

van den Oord, A., Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., Kalchbrenner, N., Senior, A. W., and Kavukcuoglu, K. (2016). Wavenet: A generative model for raw audio. *CoRR*, abs/1609.03499.

Vinyals, O., Fortunato, M., and Jaitly, N. (2015). Pointer networks. In *NIPS*.

Vinyals, O. and Le, Q. V. (2015). A neural conversational model. In *ICML Deep Learning Workshop*.

Walker, M., Stent, A., Mairesse, F., and Prasad, R. (2007). Individual and domain adaptation in sentence planning for dialogue. *JAIR*.

Walker, M. A., Rambow, O. C., and Rogati, M. (2002). Training a sentence planner for spoken dialogue using boosting. *Computer Speech and Language*.

Wang, Y.-Y. and Acero, A. (2006). Discriminative models for spoken language understanding. In *in ICSLP*.

Wang, Z. and Lemon, O. (2013). A simple and generic belief tracking mechanism for the dialog state tracking challenge: On the believability of observed information. In *Proceedings of the SIGDIAL 2013 Conference*, pages 423–432, Metz, France. Association for Computational Linguistics.

Ward, W. (1990). The cmu air travel information service: Understanding spontaneous speech. In *Proceedings of the Workshop on Speech and Natural Language*, HLT '90, pages 127–129, Stroudsburg, PA, USA. Association for Computational Linguistics.

Weizenbaum, J. (1966). Eliza&mdash;a computer program for the study of natural language communication between man and machine. *Commun. ACM*, 9(1):36–45.

Wen, T.-H., Gasic, M., Kim, D., Mrksic, N., Su, P.-H., Vandyke, D., and Young, S. (2015a). Stochastic language generation in dialogue using recurrent neural networks with convolutional sentence reranking. In *SIGdial*, pages 275–284, Prague, Czech Republic. ACL.

Wen, T.-H., Gašić, M., Mrkšić, N., Su, P.-H., Vandyke, D., Rojas-Barahona, L. M., and Young, S. (2015b). Toward multi-domain language generation using recurrent neural networks. In *NIPS Workshop of Machine Learning on SLU and Interaction*.

Wen, T.-H., Gasic, M., Mrkšić, N., Rojas Barahona, L. M., Su, P.-H., Ultes, S., Vandyke, D., and Young, S. (2016a). Conditional generation and snapshot learning in neural dialogue systems. In *EMNLP*, pages 2153–2162, Austin, Texas. ACL.

Wen, T.-H., Gasic, M., Mrkšić, N., Su, P.-H., Vandyke, D., and Young, S. (2015c). Semantically conditioned lstm-based natural language generation for spoken dialogue systems. In *EMNLP*, pages 1711–1721, Lisbon, Portugal. ACL.

Wen, T.-H., Gašić, M., Mrkšić, N., Rojas-Barahona, L. M., Su, P.-H., Vandyke, D., and Young, S. (2016b). Multi-domain neural network language generation for spoken dialogue systems. In *NAACL-HLT*, pages 120–129, San Diego, California. ACL.

Wen, T.-H., Heidel, A., yi Lee, H., Tsao, Y., and Lee, L.-S. (2013a). Recurrent neural network based language model personalization by social network crowdsourcing. In *InterSpeech*.

Wen, T.-H., Lee, H.-Y., Chen, T.-Y., and Lee, L.-S. (2012a). Personalized language modeling by crowd sourcing with social network data for voice access of cloud applications. In *SLT*.

Wen, T.-H., Lee, H.-Y., hao Su, P., and shan Lee, L. (2013b). Interactive spoken content retrieval by extended query model and continuous state space markov decision process. In *ICASSP*, pages 8510–8514.

Wen, T.-H., Miao, Y., Blunsom, P., and Young, S. (2017a). Latent intention dialogue models. In *ICML*, ICML'17. JMLR.org.

Wen, T.-H., Vandyke, D., Mrkšić, N., Gašić, M., M. Rojas-Barahona, L., Su, P.-H., Ultes, S., and Young, S. (2017b). A network-based end-to-end trainable task-oriented dialogue system. In *EACL*.

Wen, T.-H., Vandyke, D., Mrkšić, N., Gasic, M., Rojas Barahona, L. M., Su, P.-H., Ultes, S., and Young, S. (2017c). A network-based end-to-end trainable task-oriented dialogue system. In *EACL*, pages 438–449, Valencia, Spain. Association for Computational Linguistics.

Wen, T.-H., yi Lee, H., and Lee, L.-S. (2012b). Interactive spoken content retrieval with different types of actions optimized by a markov decision process. In *Interspeech*, pages 2458–2461. ISCA.

Werbos, P. J. (1988). Generalization of backpropagation with application to a recurrent gas market model. *Neural Networks*, 1(4):339 – 356.

Werbos, P. J. (1990). Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*.

Weston, J., Chopra, S., and Bordes, A. (2014). Memory networks. *CoRR*, abs/1410.3916.

Weston, J. E. (2016). Dialog-based language learning. In *NIPS*, pages 829–837. Curran Associates, Inc.

White, M., Rajkumar, R., and Martin, S. (2007). Towards broad coverage surface realization with ccg. In *In Proc. of the Workshop on Using Corpora for NLG: Language Generation and Machine Translation (UCNLG+MT*.

Williams, J., Raux, A., Ramach, D., and Black, A. (2013). The dialog state tracking challenge. In *In Proceedings of the 14th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL*.

Williams, J. D. (2010). Incremental partition recombination for efficient tracking of multiple dialog states. In *2010 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 5382–5385.

Williams, J. D. and Young, S. (2005). Scaling up pomdps for dialog management: The "summary pomdp" method. In *IEEE Workshop on Automatic Speech Recognition and Understanding, 2005.*, pages 177–182.

Williams, R. J. (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3-4):229–256.

Xia, R., Zong, C., Hu, X., and Cambria, E. (2013). Feature ensemble plus sample selection: Domain adaptation for sentiment classification. *IEEE Intelligent Systems*, 28(3):10–18.

Xu, K., Ba, J., Kiros, R., Cho, K., Courville, A. C., Salakhutdinov, R., Zemel, R. S., and Bengio, Y. (2015). Show, attend and tell: Neural image caption generation with visual attention. *CoRR*, abs/1502.03044.

Yang, Y. and Eisenstein, J. (2015). Unsupervised multi-domain adaptation with feature embeddings. In *HLT-NAACL*.

Yao, K., Peng, B., Zhang, Y., Yu, D., Zweig, G., and Shi, Y. (2014). Spoken language understanding using long short-term memory neural networks. In *IEEE SLT*.

Yin, P., Lu, Z., Li, H., and Kao, B. (2015). Neural enquirer: Learning to query tables. *arXiv preprint:1512.00965*.

Yogatama, D., Blunsom, P., Dyer, C., Grefenstette, E., and Ling, W. (2016). Learning to compose words into sentences with reinforcement learning. *CoRR*, abs/1611.09100.

Young, S. (2002). Talking to machines (statistically speaking). *ICSLP*.

Young, S. (2007). CUED standard dialogue acts: http://mi.eng.cam.ac.uk/research/dialogue/LocalDocs/dastd.pdf. Technical report, Cambridge University Engineering Dept.

Young, S., Gašić, M., Thomson, B., and Williams, J. D. (2013). Pomdp-based statistical spoken dialog systems: A review. *Proceedings of the IEEE*.

Young, S., Gašić, M., Keizer, S., Mairesse, F., Schatzmann, J., Thomson, B., and Yu, K. (2010). The hidden information state model: A practical framework for pomdp-based spoken dialogue management. *Computer, Speech and Language*.

Young, S. and Proctor, C. (1989). The design and implementation of dialogue control in voice operated database inquiry systems. *Computer Speech and Language*, 3(4):329 – 353.

Zen, H., Nose, T., Yamagishi, J., Sako, S., Masuko, T., Black, A. W., and Tokuda, K. (2007). The HMM-based speech synthesis system (HTS) version 2.0. In *Proceedings of the 6th ISCA Workshop on Speech Synthesis*, pages 294–299. ISCA.

Zen, H., Senior, A., and Schuster, M. (2013). Statistical parametric speech synthesis using deep neural networks. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 7962–7966.

Zettlemoyer, L. S. and Collins, M. (2007). Online learning of relaxed ccg grammars for parsing to logical form. In *In Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL-2007*, pages 678–687.

Zhang, X. and Lapata, M. (2014). Chinese poetry generation with recurrent neural networks. In *EMNLP*, pages 670–680, Doha, Qatar. ACL.

Zhang, X. and LeCun, Y. (2015). Text understanding from scratch. *CoRR*, abs/1502.01710.

Zhou, D. and He, Y. (2011). *Learning Conditional Random Fields from Unaligned Data for Natural Language Understanding*, pages 283–288. Springer Berlin Heidelberg, Berlin, Heidelberg.

Zue, V., Seneff, S., Glass, J., Polifroni, J., Pao, C., Hazen, T. J., and Hetherington, L. (2000). Jupiter: A telephone-based conversational interface for weather information. *IEEE Trans. on Speech and Audio Processing*, 8:85–96.