Building Blocks for the Mind

by

Jonathan Raiman

B.A., Pomona College (2013)

Submitted to the Department of Aeronautics and Astronautics
in partial fulfillment of the requirements for the degree of

Master of Science in Aeronautics and Astronautics

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

February 2015 [June 2017]

Signature redacted

Author .....
Department of Aeronautics and Astronautics
October 21, 2015

Signature redacted

Certified by .....
Brian C. Williams
Professor of Aeronautics and Astronautics
Thesis Supervisor

Signature redacted

Accepted by .....
Paulo C. Lozano
Associate Professor of Aeronautics and Astronautics
Chair, Graduate Program Committee

Building Blocks for the Mind

by

Jonathan Raiman

Submitted to the Department of Aeronautics and Astronautics
on October 21, 2015, in partial fulfillment of the
requirements for the degree of
Master of Science in Aeronautics and Astronautics

Abstract

Successful man-machine interaction requires justification and transparency for the behavior of the machine. Artificial agents now perform a variety of high risk jobs alongside humans: the need for justification is apparent when we consider the millions of dollars that can be lost by robotic traders in the stock market over misreading online news [9] or the hundreds of lives that could be saved if the behavior of plane autopilots was better understood [1]. Current state of the art approaches to man-machine interaction within a dialog, which use sentiment analysis, recommender systems, or information retrieval algorithms, fail to provide a rationale for their predictions or their internal behavior.

In this thesis, I claim that making the machine selective in the elements considered in its final computation, by enforcing sparsity at the Machine Learning stage, reveals the machine's behavior and provides justification to the user. My second claim is that selectivity in the machine's inputs acts as Occam's Razor: rather than hindering performance, enforcing sparsity allows the trained Machine Learning model to better generalize to unseen data.

I support my first claim concerning transparency and justification through two separate experiments that are each fundamental to Man-Machine interaction:

- Recommender System: Interactive plan resolution using Uhura and user profiles represented by ontologies,
- Sentiment Analysis: Text climax as support for predictions.

In the first experiment, I find that the trained system's recommendations agree better with human decisions than existing several baselines which rely on state of the art topic modelling methods that do not enforce sparsity in the input data.

In the second experiment, I obtain a new state of the art result on Sentiment Analysis and show that the trained system can now provide justification by pinpointing climactic moments in the original text that influence the sentiment of the text, unlike competing approaches.

My second claim about sparsity's regularization benefits is supported with another set of experiments, where I demonstrate significant improvement over non-sparse baselines in 3 challenging Machine Learning tasks.

Thesis Supervisor: Brian C. Williams
Title: Professor of Aeronautics and Astronautics

Acknowledgments

The work in this thesis would not have been possible without the incredible support of my advisor, Prof. Brian C. Williams. I would like to thank my collaborators and friends Peng Yu and Szymon Sidor, for their friendship, contributions, advice, and wisdom, during this research. I would also like to thank my parents, Olivier and Nathalie Raiman, and grand-father, Jacques Raiman, for their love, unrelenting support, inspiration, enriching conversations, and thought-provoking questions. I am very grateful to the entire MERS lab for the feedback and discussions, their scientific and exploratory spirit, and the creative environment they help foster. Finally, I wish to thank Prof. Regina Barzilay for the exciting discussions and her advice, along with her research group for the opportunities to share my work and talk about the future of language and learning.

THIS PAGE INTENTIONALLY LEFT BLANK

Contents

Contents 9

List of Figures 13

List of Tables 17

1 Introduction 21

1.1 Motivation 22

1.2 Problem Statement 23

1.3 Approach Overview 24

1.3.1 Ontological User Profile 25

1.3.2 Occam's Gates 28

1.3.3 Main Results 30

1.4 Related Work 32

1.4.1 Temporal Constraint Relaxation 32

1.4.2 Recommender System 33

1.4.3 Sentiment Analysis 35

1.4.4 Recurrent Neural Network Regularization 37

1.4.5 Question Answering 38

2 Ontological Recommender 43

2.1 Approach 44

2.1.1 Architecture 44

2.1.2 Reader Module 45

2.1.3 Semantic Memory 47
2.1.4 Semantic Relaxation 52
2.2 Experiment 57
2.2.1 Semantic Memory 57
2.2.2 Experimental Protocol 60
2.2.3 Evaluation 60
2.3 Results 61
2.4 Analysis 61
2.4.1 Semantic Memory 62
2.4.2 Semantic Relaxation 62

3 Sentiment Analysis **63**
3.1 Approach 63
3.1.1 Network Architecture 64
3.1.2 Sentiment Analysis Network Architecture 64
3.1.3 Vocabulary Expansion 66
3.2 Experiment 67
3.2.1 Training 68
3.2.2 Evaluation 68
3.3 Results 69
3.3.1 Comparison 69
3.3.2 Visual Explanation 71
3.4 Analysis 73
3.4.1 Memory Gate 73
3.4.2 Vocabulary Expansion 74
3.4.3 Interpretability 74
3.4.4 Comparison 75
3.4.5 Conclusion 76

4 Occam's Gates **77**
4.1 Approach 78

4.1.1 Gated LSTMs 79
4.1.2 Training Regimens 82
4.2 Experiment 83
4.2.1 Sentiment Analysis 83
4.2.2 Paraphrase Detection 84
4.2.3 Question Answering 85
4.3 Results 86
4.3.1 Effects on performance 86
4.4 Analysis 89

5 Discussion and Future Work **91**
5.1 Conclusions 91
5.1.1 Ontological User Profile 91
5.1.2 Occam's Gates 92
5.2 Future Work 93
5.2.1 Ontological User Profile 94
5.2.2 Occam's Gates 94

Bibliography **95**

THIS PAGE INTENTIONALLY LEFT BLANK

List of Figures

1-1 The Uhura Man-Machine planning system that communicates back human-understandable plan modifications, relaxations, to the human actor in best first order. 25

1-2 The Semantic Relaxation Uhura Man-Machine planning system that extends the present Uhura to include a Semantic Memory. This memory stores in an ontology user preferences and suggests, during plan relaxations, alternate options. 26

2-1 Reader Module: a shallow neural network where a word window and the distributed representation for an item serve to predict the metadata for the item. 45

2-2 An example graph S with 5 vertices, including 3 leaves. 49

2-3 With the initial observation that the green node is 1, the most likely state for the remaining leaves is computable from the joint probability derived from the Laplacian. 50

2-4 Hierarchical Cluster using the learnt distance of a subset of the restaurants collected. Here we note that the bottom right contains a mexican subtree (with its root marked with a double circle), while all vegetarian restaurants (*Veggie Grill,Loving Hut*, and *Plum Vegan Bistro*) are part of the same subtree. 51

2-5 Reserving space for semantic relaxation using an additional search node 56

2-6 Resolving conflict using temporal relaxation 56

2-7 Resolving conflict using semantic relaxation 56

2-8 2D projection obtained via T-SNE[99] of the distributed representation for all Seattle restaurants present in Yelp. The clusters in the figure correspond to places with semantically similar cuisines or with overlapping vocabulary in their reviews. 58

2-9 A second 2D projection obtained via T-SNE[99] of the distributed representation for all Seattle restaurants present in Yelp. In this figure, the intensity of the color of the dots corresponds to the log occurrence of the word "spicy". The bottom half of the learnt embedding space correlates with the presence of spicy food, a feature only made available through review text. 59

3-1 A bidirectional LSTM that uses input gates to control what subset of the input is passed on for later processing. 65

3-2 The sentence "fast but dull" is passed to a bidirectional LSTM. The forward and backward states of the network are processed by a left to right reader LSTM ($s_{1,2,3}$). The reader produces predictions $\tilde{t}_{1,2,3}$ which are gated and fused with the previous predictions $t_{0,1,2}$. 66

3-3 Root accuracy initially drops as a function of sentence length, and steadies around 45%. 70

3-4 Root accuracy drops dramatically with the number of unknown words in a sentence (apart from outliers at 8 and 10). 70

3-5 An example of MB switching emotional state from positive to negative with sentence progression 71

3-6 Sentiment Analysis over text in a separate domain. Words between asterisks were not seen during training, yet the network still reacts appropriately. 72

3-7 Sentiment Analysis over text in a separate domain. The words "desolation" and "Armstrong" were not seen during training, yet the name is kept neutral and desolation is detected as being negative 73

3-8 Sentiment Analysis over text in a separate domain. The long sentence and lack of clear sentiment markers causes indecision in the network reflected by the oscillation around neutral. 74
3-9 Sentiment Analysis over text in a separate domain. Words outside of the training data, which carry different meaning for movie reviews, like "cold" cause the network to view the sentence as negative. 75

4-1 An RNN with gates g_1, g_2, g_3 controlling the intensity of the inputs. . 79
4-2 English sentence sequence forecasting using Recurrent Neural Networks 81
4-3 A Sequence to Sequence Transducer 81
4-4 SST Root Accuracy with varying LSTM hidden size and sparsity penalty λ 88
4-5 Effect of sparsity regimen and sparsity penalty λ on SST Root Accuracy. 88
4-6 Paraphrase accuracy with varying LSTM hidden size and sparsity penalty λ 88
4-7 Effect of sparsity regimen and penalty λ on Paraphrase prediction. . . 88
4-8 Accuracy for three bAbI tasks with varying λ_{word} 88
4-9 Effect of λ_{word} and λ_{fact} on Basic Induction task accuracy 88
4-10 Gated LSTM viewing characters individually. Word boundaries are detected as shown by the absence of yellow highlighting in the inter-word spaces. 88
4-11 Gated LSTM viewing characters individually. Model focuses on upper case characters and ignores repeats. 88
4-12 Gating procedure learnt by Gated LSTMs during training for bAbI task. We note here how the yellow word highlighting in the right-most paragraph correspond to informative words, while in the leftmost paragraph no particular pattern is present. The text color indicates whether an entire sentence was kept using the fact gate. In the middle paragraph, sentences without the character mentioned in the question are already ignored. 89

THIS PAGE INTENTIONALLY LEFT BLANK

List of Tables

2.1 Comparison of model agreement with majority of human judges. . . . 61

3.1 Comparison of Root Accuracy on the *Stanford Sentiment Treebank* . . 69

4.1 LSTM and Gated LSTM equations. Modifications shown in red. . . . 80

4.2 Comparison of test accuracy using different models on bAbI dataset for various tasks from [108]. Models are (left to right): LSTM baseline from [108], Stacked Gated LSTMs, and Memory Networks. Best results for LSTMs shown in bold. 87

THIS PAGE INTENTIONALLY LEFT BLANK

List of Algorithms

2.1 Function ExpandOnConflict 55

THIS PAGE INTENTIONALLY LEFT BLANK

Chapter 1

Introduction

A common pitfall of existing Machine Learning algorithms that support Man-Machine interaction is their lack of justification and transparency. In this thesis, I claim that enforcing sparsity at the Machine Learning stage of the Artificial Intelligence collaborative system provides justification and greater transparency. I also claim that making the system's inputs sparse acts as a regulariser for Machine Learning algorithm, allowing the trained system generalize better to unseen data.

My thesis is structured as follows: in this introductory chapter I motivate the use of collaborative Man-Machine systems, next I provide a definition of the Man-Machine interaction problem within this thesis, then provide an overview of my approach and summarise my results. In the end of the chapter, I introduce current state of the art approaches to this problem and where they fall short on providing transparency and justification for their behavior.

In the second chapter I describe in further detail the problem of Recommendation within Man-Machine interaction, and present my approach. Next, I explain the experiment on which I compare agreement between human recommendations and those made by my system and several baseline systems, and summarise my results.

In the third chapter, I present my approach to Sentiment Analysis, and I describe the model's architecture and the details of the experiment. I then summarise my results by comparing the proposed approach to several ablations and the current state of the art.

In the fourth chapter, I introduce a set of experiments that support my second claim about sparsity improving generalization performance. Here, I describe in greater detail how sparsity can be systematically applied to existing state of the art approaches for Natural Language Processing, and I present three separate benchmark tasks: Sentiment Analysis, Paraphrase Detection, and Synthetic Question Answering, on which I show improved performance over several baselines. I conclude this chapter with a discussion of results.

In the fifth and final chapter, I summarise my findings and discuss areas of future work.

1.1 Motivation

Collaboration between Man and Machine is a fundamental pillar of Artificial Intelligence and an integral part of many industrial and commercial applications of intelligent agents. A key factor in the success of these systems is the ability for the human and machine to communicate: in machine assisted stock trading [9], path planning [115], vehicle fleet management [2], manufacturing [14], airborne or underwater surveys [13, 73], content curation [30, 31, 60, 3], or content summarization [5, 111], interaction between the human and Artificial Intelligence can help diagnose issues early on, replan, or consider more options than either actor could on their own.

Several obstacles remain for effective collaboration between the two actors: current Machine Learning algorithms fail to provide justification for their behavior, thereby removing crucial feedback for the human participant, while over-solicitation of the human has been shown to lead to information overload and lower performance [68, 38, 39]. Successful collaboration will require the machine to strike a balance between transparency and abstraction.

1.2 Problem Statement

> A mixed-initiative system is one in which both humans and machines can make contributions to a problem solution, often without being asked explicitly.
>
> *Jaime Carbonell, Sr.* [15]

The Man-Machine interaction problem in this thesis is as follows: both actors participate in solving a particular planning or decision-making tasks in a mixed-initiative manner [15, 69, 2].

The inputs in this problem are a series of higher-level goal states specified by the human participant:

- "I want to have dinner and be home by 8PM," or
- "What is the sentiment in the sentence: *fast but dull* ."

During the planning phase of the problem, the man and the machine exchange information either by requiring additional details from the human:

- "Would you prefer Italian or Japanese food this evening?",

or by making the machine describe its decision process:

- "because you enjoyed *Lemongrass* yesterday, and *Wild Ginger* is also asian-fusion and spicy, I recommend you go there," or
- "*fast* is positive, *but* interrupts this emotion, and finally *dull* renders the sentence negative."

The final output of the collaborative system can be a plan that can be passed to an executive, a classification label among several classes, or a sentence in Natural Language:

- "Leave in 15mn, arrive by 7PM at *Wild Ginger*, eat there, leave at 7:45PM and be home by 8PM," or
- "The sentiment is *negative* with probability 0.8."

In summary, the Man-Machine interaction problem is defined here as a collaborative effort between a human and a machine to analyze task relevant information: as input the machine receives contextual information such as a user profile or map, along with a natural language description of the desired goal state. During the analysis stage, the machine describes its decision process and can ask for additional information from the human using a dialogue or graphical interface. Finally, after analysis, a plan that can be given to an executive, a classification label, or Natural Language response is produced by the machine.

The specific problem I aim to address within this thesis is the lack of transparency during the decision process of the machine. As a remedy, I introduce a User Profile that has as its central representation an ontology, and a Machine Learning training strategy to produce models that give a visual description for the parts of their input they attend to, letting the inference process of the machine be human understandable.

1.3 Approach Overview

In this thesis, I introduce two key pieces to add transparency to the machine's decision process during the Man-Machine collaboration problem:

- **Ontological User Profile**: a more expressive representation of user preference based on an ontology of the options available to a user.
- **Occam's Gates**: a method for enforcing sparsity in the inputs to a Recurrent Neural Network which leads to an Occam's Razor behavior in the trained system.

In this section, I will first describe in greater detail how these two pieces are achieved and integrated within a mixed initiative system, and second summarize my main results.

1.3.1 Ontological User Profile

To achieve greater transparency during collaborative planning with a machine, I add an Ontological User Profile. This type of User Profile captures a human's preferences in a hierarchical manner to encode information about inheritance, entailment, and separability.

In order to describe the benefits of this addition, I will first describe the previous behavior and architecture of the mixed initiative planning system (before I add the new User Profile) and second explain how the User Profile, in combination with a semantic candidate generator, provides greater transparency and expressivity in the collaboration.

1.3.1.1 Present Uhura

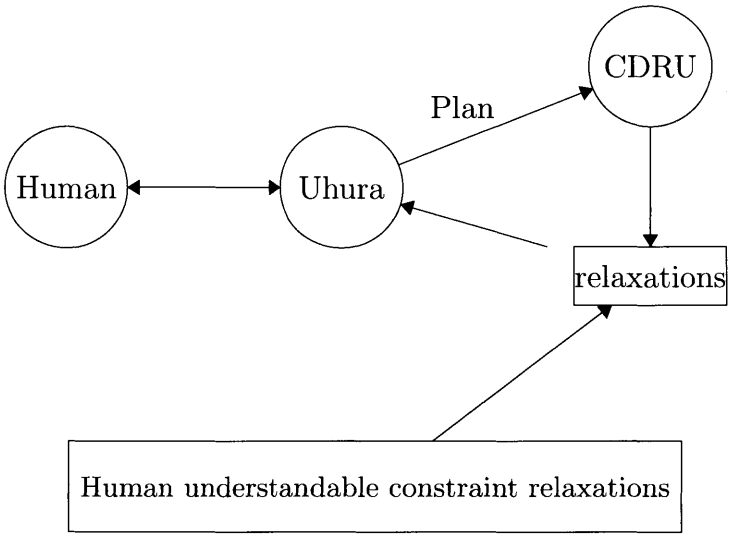

Figure 1-1: The Uhura Man-Machine planning system that communicates back human-understandable plan modifications, relaxations, to the human actor in best first order.

The goal of the Man-Machine planning system Uhura [14] is to collaboratively prepare a plan, and interactively relax constraints until the plan is temporally consistent. In order to suggest possible relaxations to the human actor, the planner makes use of the Conflict Directed Relaxation with Uncertainty algorithm (CDRU) [114]. This

algorithm resolves over-constrained temporal problems by finding relaxable temporal constraints that can be modified to restore plan consistency. The relaxations are suggested to the human in best first order by using a conflict-directed search strategy in CDRU that is similar to Conflict Directed A* [113]. The CDRU algorithm is integrated in Uhura through a dialogue system that receives the human's original plan and sends back the plan relaxations given by the conflict-directed search (**Figure 1-1**).

1.3.1.2 Semantic Relaxation Uhura

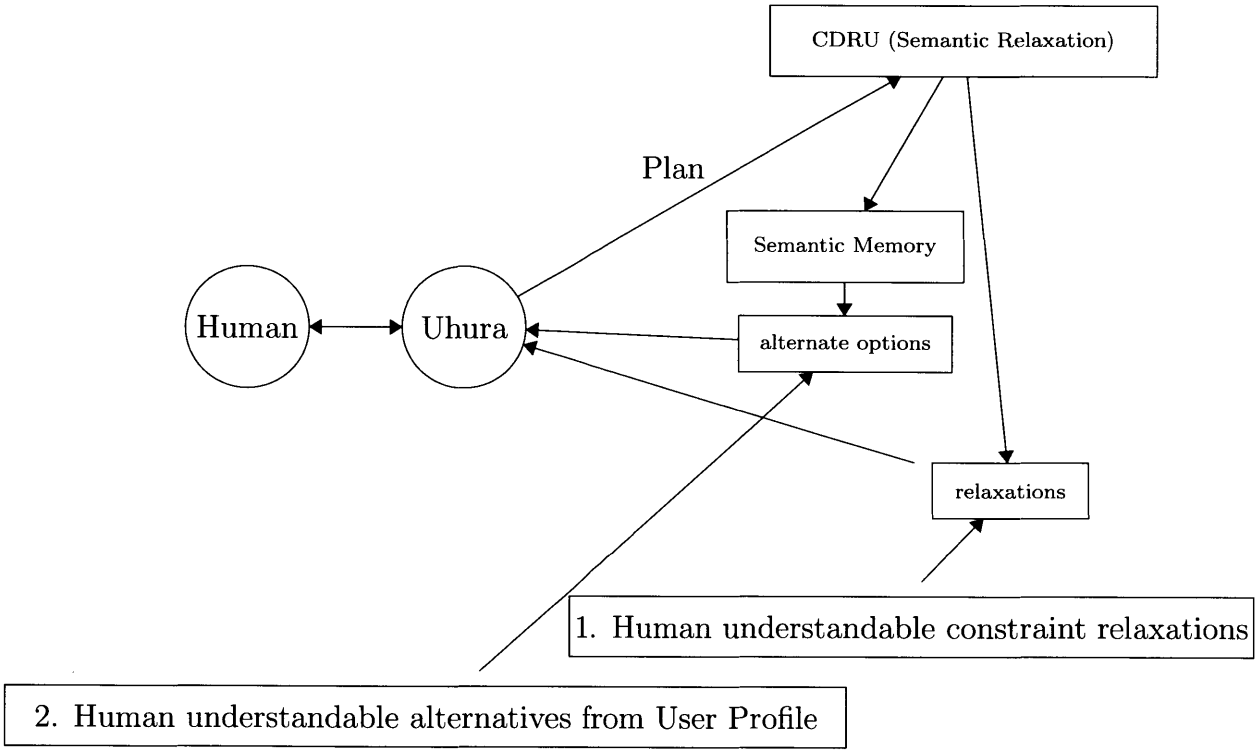

Figure 1-2: The Semantic Relaxation Uhura Man-Machine planning system that extends the present Uhura to include a Semantic Memory. This memory stores in an ontology user preferences and suggests, during plan relaxations, alternate options.

I add an Ontological User Profile to Uhura to achieve greater transparency and expressivity in a Man-Machine planning system. This User Profile gives the system an ability to describe in human-understandable terms how alternative options were suggested by the planning system and perform recommendations over new domains,

even in the absence of prior user information.

The addition of the User Profile modifies the architecture to include a Semantic Memory as visible in (**Figure 1-2**). This change has two significant effects:

- **Profile Learning**: The user's responses during interaction allow the machine to learn a model of the user's preferences with human understandable features, and
- **Ranking and Domain Extension**: The User Profile gives the Uhura system more expressivity and power by allowing ranking of options based on the learnt preferences and an ability to carry over this learnt metric to new choices discovered outside of the original plan, for instance to choices found by crawling online reviews.

1.3.1.2.1 Profile Learning

To obtain human-understandable features in a User Profile, responses from the user to the relaxation and alternate options concerning a choice y are stored with the choice's corresponding node n_y in the user's ontology. The ontology can be used to perform inference that incorporates hierarchical information about the options, and also to describe the decisions of the machine using previous user actions as supporting examples.

1.3.1.2.2 Ranking and Domain Extension

With a model of user preferences, it is now possible to rank the candidate relaxations proposed by CDRU not only according to their intrinsic cost in terms of deviation from the original plan, but also according to the agreement between a candidate solution and the best estimate of the user's preference ontology. This metric allows the Uhura system to rate higher choices that share the same parents or neighbors as those previously approved by the user.

The memory's "semantic" aspect enables the planning system to dynamically add new choices to a plan and provide a metric to compare these previously unseen choices

to those considered by the user so far. I am able to compare choices semantically through the use of a a new topic model I introduce. The topic model takes review text and available metadata and produces a vector representation for each choice. The distance between the resulting vectors can then be used to detect topical or stylistical similarity between choices.

Moreover, the topic model can be learnt once and applied later to new review text without retraining. The learnt distance between vectors is still meaningful when applied to vectors produced for late-comers: the system is therefore capable to incorporate new choices online. This modification removes the restriction that the relaxations proposed by the CDRU algorithm be contained in the original temporal problem: choices can now be dynamically instantiated from online reviews, and embedded using the same topic modelling algorithm used initially.

1.3.2 Occam's Gates

Occam's Gates, an objective function to enforce sparsity in the inputs of a Recurrent Neural Network, is the second piece I propose to add transparency in the collaboration between Man and Machine. Learning from sequential data, such as speech or text, is at the heart of many communication and interaction tasks in mixed initiative systems. Recurrent Neural Networks (RNN) have emerged as a very powerful family of models that are able to process and keep track of long term dependencies in sequential data, enabling machines to detect temporal patterns in speech, language, or actions.

I modify the objective function used to train RNNs to include gates controlling information flow to the network, along with a penalty on the activation of the gates to encourage selectivity in the network. The trained networks can then produce a visual explanation for the temporal regions that were attended to, giving insight into the information flow in the system.

1.3.2.1 Occam's Gates Operation

To understand how Occam's Gates improve the transparency of machines that use RNNs as sequence learning machines, I will first describe briefly their operation, before defining more formally my approach.

Definition 1.1. Recurrent Neural Network (RNN)

A Recurrent Neural Network is an artificial neural network with an ability to update its internal state. The internal state is modified by connections between neuronal units that form a directed cycle. These networks are defined by a state update mechanism that operates at every timestep t in a sequence of inputs $\{x_1, \ldots, x_n\}$, which we denote $f_{\text{RNN}}(\cdot)$.

The state update mechanism takes as input the previous state h_{t-1}, and the current input x_t, to produce a new state h_t:

$$f_{\text{RNN}}(x_t, h_{t-1}) = h_t.$$

In order to make the behavior of the RNN more transparent, I impose a sparsity penalty on the amount of information the network used at every timestep t. To control the flow of information to the RNN, I use a gating function that takes the previous RNN state and current input and produces a scalar value in the range $0-1$: $g(h_{t-1}, x_t) = g_t$. The scalar gating value, g_t, is element-wise multiplied with the input at the current time step x_t, allowing the network to ignore with $g_t = 0$, or attend to $g_t = 1$, a timestep's input.

The new gated state update function for an RNN, with a state update function $f_{\text{RNN}}(\cdot)$, now becomes:

$$f_{\text{updated}}(x_t, h_{t-1}) = f_{\text{RNN}}(g(x_t, h_{t-1}) \odot x_t, h_{t-1}).$$

The original training objective for the overall network, J, is augmented to enforce sparsity in the outputs of the gating function g through a sparsity penalty weight λ_{sparse}:

$$J^* = J + \lambda_{\text{sparse}} \cdot \sum_i g_i. \tag{1.1}$$

1.3.2.2 Occam's Gates for Justification

Once trained, an RNN that uses Occam's Gates now produces gating values, $\{g_1, \ldots, g_n\}$, for every timestep of an input sequence, $\{x_1, \ldots, x_n\}$, which can be used in several ways to make the behavior of the machine more understandable to the human actor:

- The gating values act as a highlighter, pinpointing temporal regions where information from the input affected the network's predictions and behavior.
- The gating values can serve to subsample a long input stream to filter out unnecessary or superfluous portions to form a summary sequence.
- The ratio of active versus inactive gates describes in qualitative terms the signal to noise ratio of a sequence.
- The firing location of the gates serves as an early warning signs for overfitting and other issues with a network: for instance, when these locations no longer correlate with information units found by humans.

1.3.3 Main Results

In my experiments, I validate the effectiveness, gained transparency, and added performance of the Ontological User Profile and Occam's Gates.

1.3.3.1 Ontological User Profile

I compare the recommendations made by Uhura with an Ontological User Profile with those made by other topic modelling approaches and find that the proposed approach produces better agreement with human judges. I make this observation in an experiment where Mechanical Turk workers must select among several restaurant

options $\{o_1, \ldots, o_n\}$, the best one to recommend for a person who likes several other restaurants $\{r_1, \ldots, r_k\}$. I compare the decisions made by the workers with the recommendations made by Uhura by initialising the user profile with the restaurants $\{r_1, \ldots, r_k\}$, and ranking the options $\{o_1, \ldots, o_n\}$. I compare the Ontological User Profile in Uhura with several state of the art topic modelling strategies, which do not make use of the hierarchical nature of the choices, or cannot combine the review data with the metadata found online. I find that using an Ontological User Profile built from online reviews improves agreement with humans by 6.4% (from 36.11% to 43.52%) over competing approaches.

1.3.3.2 Occam's Gates

The justification process used in Occam's Gates has several key findings and outcomes for sequence learning tasks and transparency in the behavior of the machine.

1.3.3.2.1 Sentiment Analysis

Representing sentiment as a gated, justifiable process, with each timestep's prediction updating the current best prediction rather than replacing it, improves prediction accuracy and provides a visual description for the areas of a text containing emotional signal. A model trained with the gating procedure outperforms a baseline by 7.03% on 5-class accuracy and 5.73% on binary accuracy, and exceeds the current state of the art by 1.3% on 5-class accuracy, and 0.83% on binary accuracy.

1.3.3.2.2 Sequence Learning

Occam's Gates (OG) are also systemically applicable to other problems; I compare several baseline architectures to gated ones. The gated systems trained with OG consistently outperform their non-sparse baselines:

- **Sentiment Analysis**: Models trained with OG improve by 5%, with the performance difference growing with model size. This ability to train larger models

suggests that OG acts as an implicit regularizer by communicating the problem structure.

- **Paraphrase Detection**: Models trained with OG improve by 18% on Paraphrase Prediction recall, and follow a similar trend with respect to model size than the one seen in Sentiment Analysis.

- **Question Answering and Fact Reading**: (bAbI dataset) The performance of LSTMs with OG improves on 17 out of the 20 tasks. For some tasks, the performance of LSTMs with OG now matches that of the more sophisticated Memory Networks, simply through the addition of a sparsity penalty.

1.3.3.3 Summary

In summary, I find that enforcing sparsity either through a User Profile that incorporates hierarchical knowledge about the options, or at the Machine Learning stage by training Recurrent Neural Networks to be sparse in their use of input data, or selective in their updates to their predictions, adds transparency to the machine's behavior, provides better agreement with human behavior, and improves the performance on sequence learning tasks, and establishes a new state of the art result in Sentiment Analysis.

1.4 Related Work

Man-Machine interaction is a well studied problem. Here, I provide an overview of existing approaches to this problem and describe how they relate to the work presented in this thesis.

1.4.1 Temporal Constraint Relaxation

Joint planning of actions between a human and a machine requires the ability to construct, modify, and detect problems within plans (*temporal problems*). The work presented in this thesis builds upon an existing system with this capability, Uhura [14].

Uhura uses the Conflict Directed Relaxation with Uncertainty algorithm, (CDRU, [114]) to recover temporal consistency in *over-constrained temporal problems* with uncertain durations: whenever no execution strategy [100] or event schedule [21] can be found that satisfies all its constraints. The CDRU algorithm finds constraints within the over-constrained problem that can be continuously relaxed or removed in order to recover temporal consistency.

Unfortunately, Uhura and other approaches to joint planning between man and machine only consider a static number of temporal options as changes to recover consistency within the plan. Here, I present an approach that adds a candidate generator to Uhura to provide alternate options that resolve consistency through the use of *Semantic Relaxation.*

This candidate generation algorithm extends the original work on temporal constraint relaxation, through the use of a user profile as a candidate generator for alternate solutions. The user profile is learnt through previous constraint relaxations and dialogue with a human. The history of the user profile serves to rank and propose new options to the relaxation algorithm. This modification to the system allows it to consider a dynamic number of options to recover consistency, thereby increasing the robustness and flexibility of the plan relaxation to deal with new and unexpected situations.

1.4.2 Recommender System

The candidate generation algorithm introduced in this thesis to support *Semantic Relaxation* in Uhura relies on a Recommender System to generate, update, and use a user profile.

In a typical setting, Recommender Systems rely on user preferences or action history to rank different options, however in the case of planning, the choices are generated on the fly. Moreover, due to the use of a candidate generator in Uhura, the recommender system must be able to rank new options as well. The work presented in this thesis thus combines research from two different types of recommender systems:

- *Warm Start*, where past knowledge about user behavior is available to the recommender system, or
- *Cold Start*, where little or no past user behavior exists: the system learns from the ground up.

The choices shown to a human actor in the collaboration form a mix of *warm* and *cold* start situations.

1.4.2.1 Warm Start

Ensemble techniques making use of matrix factorization has shown great success in the presence of extensive prior user data [82, 91, 7].

In cases with more limited recommendation domains, where finer granularity in the preference of users is required, the Conditional Preference [12] or Probabilistic Conditional Preference [17] representation allows inference while making use of more advanced rules about user behavior.

1.4.2.2 Cold Start

In cold-start situations where no or limited prior user data is available, techniques which make use of item-to-item similarity are used [70, 83, 91]. The user profile in these instances is constructed by grouping or weighing the representations for the items that the user interacts with.

Also relevant to cold-start recommendation is research on infant concept learning, a subject of significant research within the Cognitive Science field [50, 96, 97, 98], which can be reformulated to rank different choices in a Recommender System without requiring prior user information.

The original goal of the infant concept learning body of work is to understand the generative model or "theory" that governs the actions of an infant or human. By reformulating this work for use in a Recommender System, it is possible to rank different "theories" that explain a user' preferences: in *The Discovery of Structural Form*, Tenenbaum and Kemp demonstrate that different underlying topologies for

describing observation data lead to different predictive models for human behavior. In the case of categorical information, such as plants, animals, and other entities that share traits in a genealogical manner, the authors find that representing observations about the world using an ontology better correlates with human behavior.

1.4.2.3 Ontological User profiles

Current Recommender Systems either rely exclusively on user-to-user information, or item-to-item similarity, while those that combine both pieces of information do not make use of the ontological structure of the items.

Recent work on a hybrid matrix factorization and item-to-item similarity models [54] has been shown to dynamically give the best of both *cold* and *warm* start systems depending on the availability of user data, however this system cannot make use of the categorical or ontological relationships between items, which play a critical role in inferring human behavior [50].

In this thesis, I present an extension to the work of Kemp et al. for use in recommender systems, which allows hybrid recommender systems to leverage structural information, while also overcoming the need for a predefined ontology. I provide a methodology for representing user preferences within the framework of *The Discovery of Structural Form* and ranking options, and finally introduce a new topic modelling algorithm that permits the generation of ontologies using only web-scraped data containing reviews for the items under recommendation.

1.4.3 Sentiment Analysis

A key component in dialogue systems is the ability for the machine to detect and react to the user's emotional state. There exists several channels through which a machine can attempt to predict the user's emotional state: voice [19], facial expressions [32], or diction [87]. In this thesis I focus on detecting emotional state through diction by analysing text.

Early approaches to predicting the polarity of a body of text relied on n-gram

and other hand engineered features of phrases and obtained strong results over longer texts [71] . However these techniques' performance severely degrades when the text is short, has spelling or grammatical mistakes, or uses slang and shorthand [87, 107].

1.4.3.1 Tree, Recurrent, and Convolutional Neural Networks

An alternative representation of text and sentences, which raised the binary classification accuracy to 88% from 60%, was the use of syntactic parse trees to shape the architecture of a neural network [86, 87, 43, 42, 40, 41]. In the work of [65, 59], the authors make specific mention of quadratic forms as instrumental for capturing the effects of negation and reinforcement in human language within syntactic parse trees.

Following the work on syntactic parse trees, several other successful representations of sentences for sentiment analysis have emerged including convolutional neural networks applying filters over the temporal domain of a sentence, using recurrent neural networks, or even using recurrent neural networks over a tree (Tree-LSTM) [94], with the ability to control how the information from multiple children arriving at a node is combined.

1.4.3.2 Attentional Networks

Finally the *Dynamic Memory Network* [55] is particularly relevant to the framework I introduce here: in their approach, each word in a sentence gets processed by a gate which selects what to include into its *episodic memory.* This external memory structure permits the network to perform multiple passes through the text before making a prediction, allowing for an iterative solution to emerge. The authors of [55] compare their technique to several baseline RNN, such as stacked-LSTMs, and find that the *episodic memory* provides an additional 1.5% accuracy on fine-grained sentiment classification. This result supports the claim that attentional neural networks can construct or mimic the effect of syntactic parse trees without requiring this information during training.

1.4.3.3 Justification

Current state of the approaches do not provide a way of understanding the underlying decision process of the machine when estimating the sentiment of a text. While the work on attentional neural networks demonstrated in [55, 94] gives the system an ability to focus and control a memory structure, the gating mechanisms do not map directly to the prediction task: the area of attention of the network is not human-understandable.

There has been effort towards making the focus of a neural network human-understandable for Sentiment Analysis, as in the work of [45], however their approach is achieved by post-processing and is separate from the machine's decision process. In their work, the authors trained convolutional neural network to predict the sentiment, and present a method for "un-pooling" the convolutional filters to recover which n-grams and words in a sentence affect the final prediction. While the recovered words and n-grams indicate portions that were attended to, the way by which these words or n-grams are composed is not available.

In this thesis, I introduce a method for providing justification within sequence prediction tasks through a new machine learning framework which uses intermediate predictions as part of its state. The trained system outputs the evolution of its prediction as a sequence is read in, giving a clear indication for which portion of the input were climactic and led to changes in the prediction.

1.4.4 Recurrent Neural Network Regularization

The work presented here on Occam's Gates resembles research done in Machine Learning on regularizing Recurrent Neural Networks.

RNN regularization has recently been shown to be achievable using *Dropout* [90] by regularizing a subset of the recurrent connections in deep RNNs [116, 75]. Previously, it was shown that weight decay regularization only provided small improvement [27] and dropout noise was detrimental when applied to all connections due to the compounding of errors over time [6].

In this thesis, I show that these kinds of regularization techniques can be combined and solved deterministically by penalizing gate activations within RNNs.

1.4.5 Question Answering

Within the Man-Machine interaction, both actors communicate through a series of questions and answers in natural language. Question Answering has been extensively studied in support of mixed-initiative systems. In this thesis, I build upon several techniques developed to support question answering for dialogue systems.

1.4.5.1 Factoid Question Answering

In Factoid Question Answering, the goal is to correctly select or combine previously stored facts and return them when given the matching question. Research in this domain has moved from more rigid approaches like DARPA's Airline Travel Information System (ATIS), which used slots to detect travel related information in spoken language, to sets of rules and patterns that governed a dialog in [34, 2], to grammar and regular expression rules that allowed systems to answer on a broader of topics as demonstrated through the START system [48, 49, 47][1] and the TREC question answering tracks[106].

The question answering system supporting the dialogue interface within the interactive plan resolution component of Uhura [14] uses a slot-based system to correctly match statements from a user to fields required by the machine for the task.

There exists several shortcoming with fact based question answering which I attempt to address in this thesis:

- Application domain is fixed in advance by the slots, rules, or database used.
- Responses are based on templates or are fixed.
- Slot-based techniques fail when the returned text contains misspellings or a different phrase structure than expected.[2]

[1] `http://start.csail.mit.edu/index.php`

[2] For instance, if the response contains extra information.

1.4.5.2 Fact Reading and Question Answering

More recently, there has been renewed effort into constructing systems that can answer questions over children's or real world stories by jointly training a system to collect information and respond to questions. This integrated approach aims to address several of the issues with fact-based QA by allowing adaptation to new fact databases, and producing responses without pre-defined templates.

The performance of this systems is hard to measure exactly because the set of allowable responses is greater or infinite. Several benchmark datasets have emerged as robust measures of performance on this task which narrow the vocabulary or use multiple choice responses:

- **MC Test** : this dataset [78] is a collection of children's stories aimed at a reading level of a 7 year old or lower, which assess coreference and general reasoning skills, while requiring a basic understanding of English.
- **bAbI Tasks** : Facebook's bAbI datasets [110] test a variety of skills from navigation to induction, through the use of synthetic stories with associated questions.

With the introduction of these datasets, several new models have been developed that are capable of creating and recalling memories to produce textual output, allowing them to be more robust to the limited training data and minimal supervision found there [76, 110, 11, 109, 55, 92].

1.4.5.3 End-to-End Question Answering

Finally, early attempts to training end-to-end systems capable of maintaining a conversation have shown promise at generating believable responses. In the work of [103, 89, 84], systems trained to produce or respond to dialog from movie scripts, support lines, and subtitles, can successfully use this training to engage in new dialogues. However, those systems are constructed without fixed personalities or goals, and hence tend to drift or forget earlier interactions, as the conversation goes on.

1.4.5.4 Justification

The move towards fact reading QA systems and end-to-end training has resulted in more robust systems that are able to cope with more limited training data, larger vocabularies, and make use of more advanced reasoning to solve queries, however this has also made the resulting systems more opaque. Improvements in robustness are attributed to the use of word embeddings [74, 63, 110, 10], and Recurrent Neural Networks [76, 55, 92, 103, 89, 84], both techniques that make use of distributed representations, where the precise flow of information is obscured by the high dimensionality of the representations.

There have been efforts towards making the behavior of these neuronal architectures comprehensible by humans such as the *episodic memory* found in the DMN [55], where the system highlights the facts that it is using to answer in the order they are processed, giving the human an ability to visualize the reasoning process of the machine. However, enabling visualization of the reasoning process in the DMN requires fact supervision: extra labels in the training dataset that indicate whether a fact is relevant. This type of annotation is currently only found in the bAbI dataset, and thus severely limits the ability for DMN-like approaches to output their reasoning process. Another line of work relies on a Softmax Attention model augmenting recurrent neural networks to look back upon a second series of hidden states to extend its memory, as demonstrated in [80], where this approach was successfully used to improve entailment detection using an attentional neural architecture. While this approach does not require fact supervision, it does not provide a way of controlling how sparse the model's attention should be. This lack of constraint often leads the model to produce diluted attentions that lack interpretability.

In this thesis, I present an augmented objective function that removes the need for fact supervision while still allowing the machine to provide a visual description of the information it selected in its computation of an answer. This method relies on the use of input gates called Occam's Gates that control the flow of information into the Neural Network. The addition of a sparsity penalty on the activation of these

gates to the original classification objective function now makes it possible to train a network to use these gates to follow an Occam's Razor principle: the system is taught to be selective in its choice of facts, and to output a visual description of the portions of the input that were chosen by the machine. The proposed method is general, and can be systematically applied to other sequence learning tasks, and I demonstrate its effectiveness on 3 different sequence learning tasks.

THIS PAGE INTENTIONALLY LEFT BLANK

Chapter 2

Ontological Recommender

Two significant challenges in existing mixed-initiative planning systems are extensibility and interpretability. Current state of the art approaches to recommendation of choices and options within these systems focus on the *warm start* situation, where a wealth of prior user data exists, however this information is often unavailable or difficult to keep up to date. Another challenge within current recommendation systems is the lack of transparency in their decision process.

In this chapter, I present joint work with Peng Yu on an Ontological User Profile that gives the mixed-initiative planning system, Uhura, an ability to learn from prior user experience online, reason about preferences hierarchically, justify decisions, and dynamically introduce new choices and rank them using a previously learnt user preference.

This chapter is structured as follows: in the first section I describe our approach, starting with a description of the modified Uhura system, then I introduce the Semantic Memory module that interfaces between the user profile and the plan resolution algorithm (CDRU [114]), and finally I describe how the extended system operates using *Semantic Relaxation.* In the second section, I introduce an experiment that provides empirical validation for the effectiveness of our approach by comparing the system's recommendations with humans', and in a third section I provide our results, which I analyze in a fourth section.

2.1 Approach

The introduction of an Ontological User Profile involves several modifications of the architecture of the mixed initiative system to allow profile learning, option ranking respecting their hierarchy, an ability to justify recommendations, and dynamic acquisition of new choices.

In order to describe each of these changes, I will first provide an overview of the main architectural changes before I go into further detail into each sub-component.

2.1.1 Architecture

The modified Uhura architecture adds two new components: a Reader Module, and a Semantic Memory, to a modified Planner Module (CDRU [114]).

1. **Reader Module** : The Reader Module uses a topic model to obtain vector representations of choices from their associated online reviews and metadata. The resulting vectors can be used to compare options categorically, and stylistically, or to inform the User Profile.

2. **Semantic Memory** : The Semantic Memory contains an ontology for each user the system keeps track of. The ontology is constructed as a weighted tree with the different choices available to a user as leaves. The relationship between child and parent nodes in the ontology follow an "`is-a`" relationship: neighbors and parents are thereby semantically similar. At each node a label holding the user's preference can be in three states: {likes, dislikes, unknown}. The Semantic Memory uses these labels to rank other choices and provide candidate solutions to the Planner Module during relaxation.

3. **Planner Module** : The Planner Module using a modified CDRU algorithm called *Semantic Relaxation* to recover temporal consistency in temporal problems given by the user. To recover consistency, this module performs continuous relaxation of temporal constraints, along with dynamic introduction of alternatives by querying the Semantic Memory.

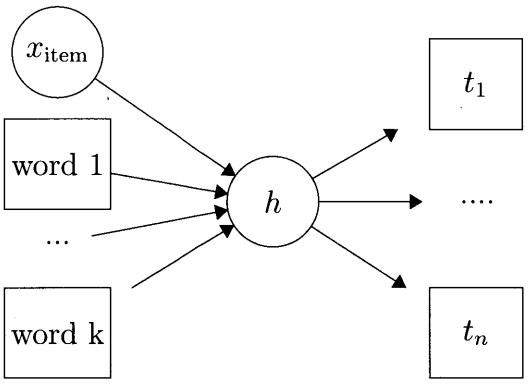

Figure 2-1: Reader Module: a shallow neural network where a word window and the distributed representation for an item serve to predict the metadata for the item.

2.1.2 Reader Module

In order to combine user behavior and raw information collected about recommendable items, our system's reader module uses a topic modelling algorithm to induce a topology over choices. As a result of this topology, categorical and stylistical similarities between choices can be used for ranking and justifying the machine's decision.

This methodology allows us to make the system more transparent by creating a meaningful distance between items, which, as we will later show in our experiments, better corresponds to human judgment than methods using explicit or implicit features separately.

An ideal recommendation system should find a replica for what a user wants if the original is unavailable. Meeting this demanding criteria is often impossible, except in the case of mass produced goods, movie sequels, or chain restaurants; in the remainder of cases, recommendation relies on understanding the underlying motivation and using this as a search criteria. It is hard to extract in an unsupervised fashion from a description of an item what is attractive about it. On the other hand, there has been success in the Natural Language Processing community on topic modelling to derive a distance that encodes the semantic proximity between documents [57, 118, 56, 33].

Inspired by these advances, we constructed our Reader Module as a shallow neural network (**Figure 2-1**) similar to the one described in [57] that learns a vector

representation for an item.

To learn this representation we perform an autoregressive task: using a randomly selected window of words, $\{w_1, \ldots, w_k\}$, from the reviews of an item, along with the vector representation of this item x_{item}, we predict the metadata $\{t_1, \ldots, t_n\}$. We project each word in the window into a word vector of dimension d, and through back-propagation we learn the parameters for x_{item} and the embedding matrix E for the words, $\{\text{Embed}(w_1, E), \ldots, \text{Embed}(w_k, E)\}$.

The parameters for this network are as follows, with k the size of the word window, n the number of possible metadata values (both binary and categorical), and x_{item} with dimensionality d_{item}: $\mathbf{W}_{\text{linear}}$ an $(d_{\text{item}} + k \cdot d) \times n$ matrix, $\vec{b}$ a bias vector with dimension n. The equations for this network are as follows:

$$y = \begin{bmatrix} \text{Embed}(w_1, E) \\ \vdots \\ \text{Embed}(w_k, E) \\ x_{\text{item}} \end{bmatrix}^T,$$

$$f_{\text{linear}}(y) = y \cdot \mathbf{W}_{\text{linear}} + \vec{b}.$$

The feed forward connections between word vectors, item representation x_{item}, and the predicted labels, can be extended to contain higher order interactions through the use of a quadratic form through the use of an additional tensor, $\mathbf{W}_{\text{quad}}$, with dimension $(d_{\text{item}} + k \cdot d) \times (d_{\text{item}} + k \cdot d) \times n$:

$$f_{\text{quad}}(y) = f_{\text{linear}}(y) + y \cdot (y \cdot \mathbf{W}_{\text{quad}})^T.$$

We define the probability of a binary label t_i being true as follows, with $\sigma(\cdot)$ the logistic function:

$$\mathbb{P}(t_i|y) = \sigma(f_{\text{linear,quad}}(y)_i).$$

For categorical labels, we define the probability of the i-th label being present in a group of m categorically exclusive labels, with indices $c_1, \ldots, c_m$ to be as follows:

$$\mathbb{P}(c_i|y) = \text{Softmax}(f_{\text{linear,quad}}(y)_{c_1,\ldots,c_m})_i.$$

Our error function in this network is the Kullback-Leibler divergence between the presence of a label and its predicted probability under the model. After training, the resulting item vectors are used as inputs to the Semantic Memory.

2.1.3 Semantic Memory

The Semantic Memory is a representation of a user profile that uses an ontology to keep track of user behavior and rank options using the discovered preference. In order to describe the operation of this module I will first provide background on its origins in *The Discovery of Structural Form*, then explain how the representation is modified for use in our recommender system, and conclude with an example.

2.1.3.1 Background

Representation of human beliefs, and in this case preferences, using an ontology was shown to be an effective representation in Cognitive Science research on Infant Concept Learning [50], when using a distance metric on graphs called the Commute Distance [105], where graph nodes are close if they share many connecting paths more than if they only have a single short path. In this sub-section, I will give some background Infant Concept Learning research relevant to the Semantic Memory, followed by an overview and example of Commute Distance applied to an ontology similar to the one used for user preferences.

2.1.3.1.1 Infant Concept Learning

Recommendation without prior user information relies on excellent knowledge transfer from item-to-item similarity to user-to-user similarity. Recommendation within this new context is now zero-shot learning, a holy grail of artificial intelligence. For-

tunately, children from a very young age are adept at this. Indeed, humans can early on distinguish their parents from others, develop speech and motor skills without strong supervision, but rather through the use of mirror neurons and an excellent generalization ability [25, 79].

In Kemp and Tenenbaum's framework, they explain how humans can generalize far away from what they have witnessed by forming theories. They hypothesize that humans have a strong prior belief on the underlying generative structure of the data they observe. In other words, each example refines an existing and carefully thought-out view of the world. They provide evidence within their work [98, 50, 96, 97] for the ability of their model to make similar predictions to those of a human when asked to fill blanks within a set of observations. For instance, in [51], the authors ask human subjects what gene an animal possesses, and they find that using taxonomy trees as the structure in their model yields high correlation ($r = 0.9$), while a diffusive or "web" structure has slightly negative correlation ($r = -0.09$). Similar observations are made about which structural form best explain how humans reason about the presence of diseases and places.

2.1.3.1.2 Commute Distance

The inference procedure described in *The discovery of structural form* [51] uses the *Commute Distance*, a metric defined on graphs useful for predictions. The Commute Distance [105] or Resistance Distance [53] is a graph metric which takes into account not only the distance between graph nodes but also the number of different paths connecting them. This metric enables the parametrization of a multivariate Gaussian from a graph using its Laplacian Δ.

Following the approach taken by Zhu et al. [120], the authors of [51] define an observation vector $\vec{y} \sim \mathcal{N}(0, \mathbf{K})$ on a graph S, with nodes N, node-degree D, adjacency matrix $\mathbf{A}$, and $\mathbf{K}$ the covariance matrix for the random variable $\vec{y}$, with σ^2 the variance for the binary observation at every node:

$$\Delta = \text{Laplacian}(S) = \text{diag}(D) - \mathbf{A} \tag{2.1}$$

$$K = \left(\Delta + \frac{1}{\sigma^2}\mathbf{I}\right)^{-1} \tag{2.2}$$

$$y \sim \mathcal{N}(0, \mathbf{K}). \tag{2.3}$$

This model enables inference over a graph and makes use of the structure to inform the covariance matrix for the random variable $\vec{y}$.

Example 2.1. *Predicting the state of tree nodes*

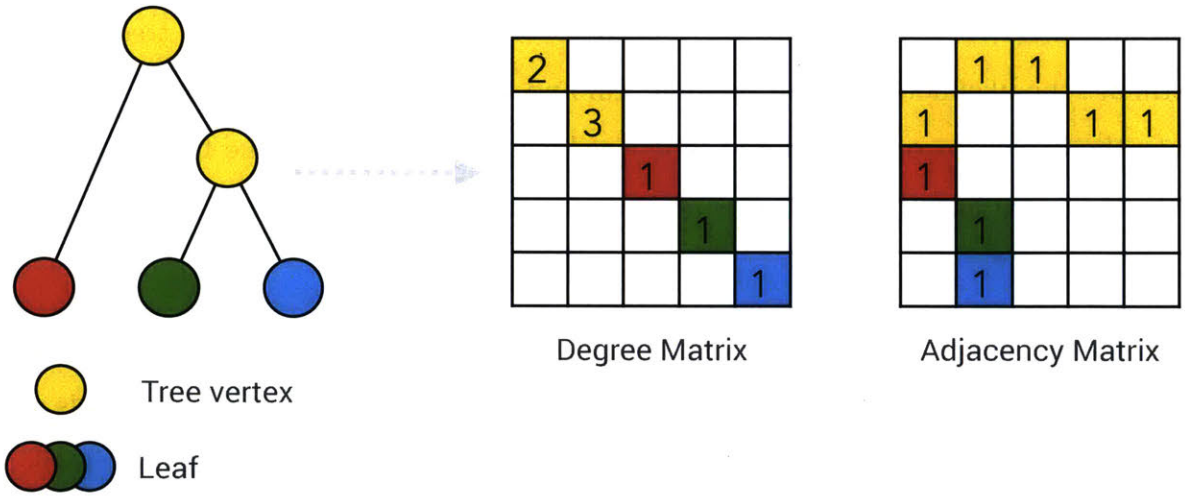

Figure 2-2: An example graph S with 5 vertices, including 3 leaves.

Let us consider the following example that demonstrates the effect of Commute Distance for making inference on simple ontology: consider a tree with 5 vertices, and 3 leaves (**Figure 2-2**). The goal is to compute the likely state l_1, l_2, l_3 of the leaves on the tree ($l_{1,2,3} \in \{0, 1\}$), given our current observation of the green leaf having value 1.

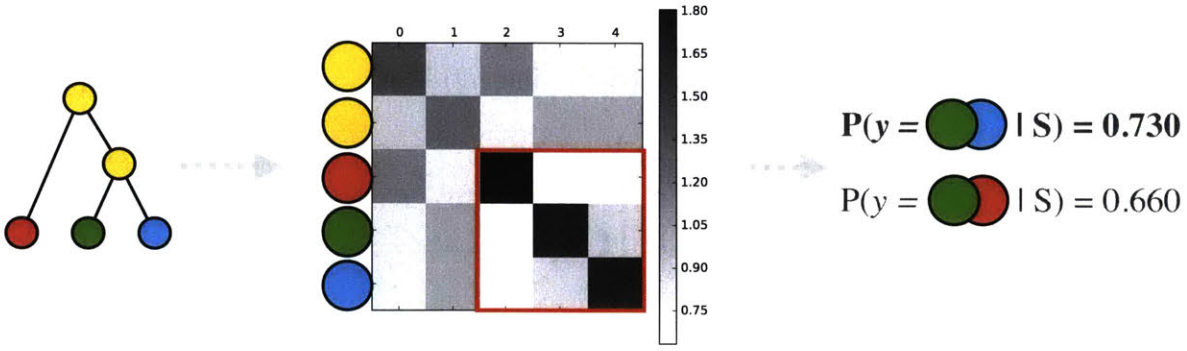

Figure 2-3: With the initial observation that the green node is 1, the most likely state for the remaining leaves is computable from the joint probability derived from the Laplacian.

Following the steps in **equation 2.1** the covariance matrix K corresponding to this tree can be obtained, shown in (**Figure 2-3**). The red box in (**Figure 2-3**) is the submatrix corresponding to the covariance between the leaves of the tree. I can estimate the likelihood of both the blue leaf and the green leaf being 1, and compare it to the likelihood of the red and blue leaf being 1. As shown in the figure, having both the green and the blue leaves be 1 at the same time is more likely.

2.1.3.2 Semantic Memory Operation

Formally, the semantic memory is defined as a weighted directed graph, with labels at every node of the tree. A subset of the nodes of the graph are recommendable items, while the remainder encode an "`is-a`" ontological relationship between nodes. At every recommendable node in the tree, the label can be in 3 states: {liked, disliked, unknown}. In the example semantic memory shown in (**Figure 2-4**), only leaves are recommendable items.

Example 2.2. *Planning a trip*

Let us consider the following example: a man and a machine are having a dialog to plan a trip in Seattle. The semantic memory in this example is illustrated in (**Figure 2-4**). In this example, the human expresses interest in going to *Loving Hut*,

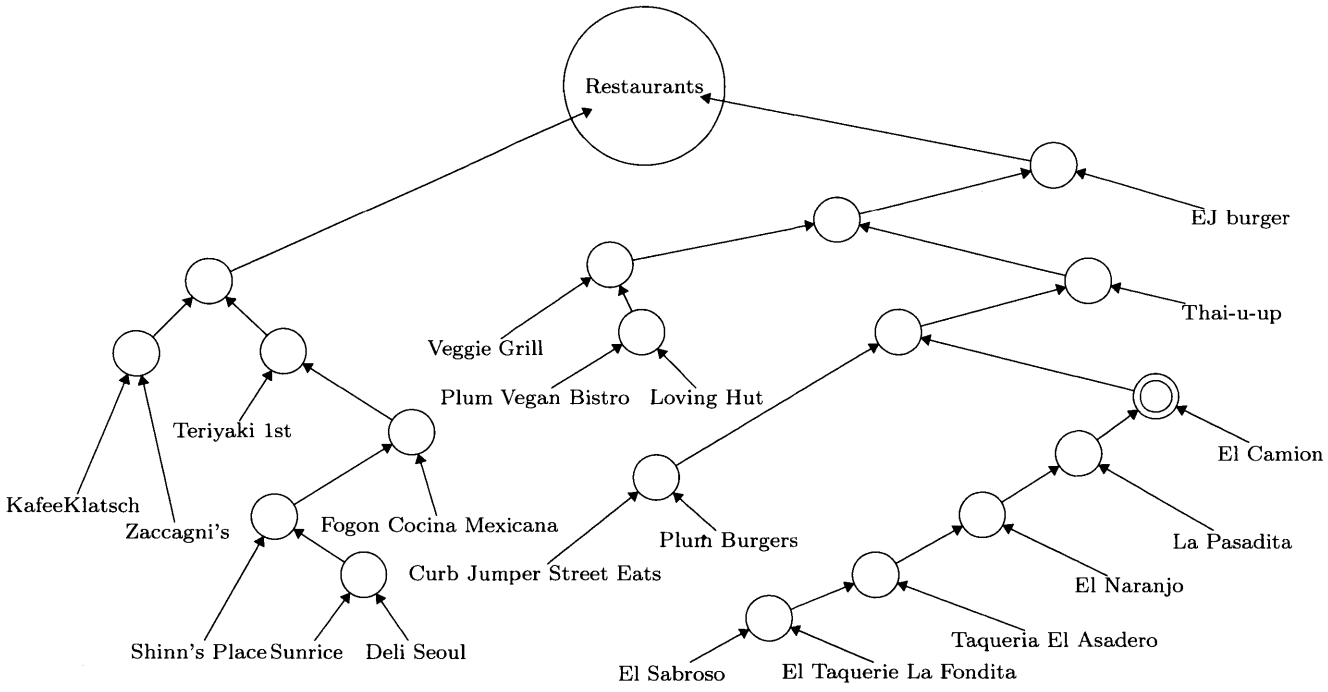

Figure 2-4: Hierarchical Cluster using the learnt distance of a subset of the restaurants collected. Here we note that the bottom right contains a mexican subtree (with its root marked with a double circle), while all vegetarian restaurants (*Veggie Grill*,*Loving Hut*, and *Plum Vegan Bistro*) are part of the same subtree.

a vegetarian restaurant. The semantic memory remembers this observation inside the tree.

During a future dialog between the man and the machine, memories concerning what restaurants were liked and disliked with affect how recommendations are ranked using the relationship between items to control the re-ranking. In this example, following the logic described in (2.1.3.1.2), *Plum Vegan Bistro* and *Veggie Grill* will be most highly ranked due to their proximity to a leaf that was observed to be liked.

In this example, let us suppose that *Loving Hut* is closed, and *Plum Vegan Bistro* is 30 minutes away from where the person is situated, while *Veggie Grill* is only 15 minutes away. If time were no object, then the semantic memory would rank *Plum Vegan Bistro* highest. If during the dialog the Planner Module learns about a temporal constraint preventing this longer trip, then the semantic memory would eliminate this option from its tree, leaving *Veggie Grill* as the top option.

2.1.4 Semantic Relaxation

Through Semantic Relaxation, the Planner Module performs constraint relaxation on a large set of options that were not encoded in the original problem and obtains candidate solutions that are highly rated using a user's profile.

2.1.4.1 Background

This component acts as a candidate generator for a temporal relaxation algorithm, Conflict Directed Relaxation with Uncertainty (BDCR). The BCDR algorithm enumerates preferred continuous relaxations to resolve over-subscribed conditional temporal problems [115].

In prior work, the BCDR algorithm was incorporated as part of a trip advisor to help rental car users to make decisions between alternative trip destinations, duration of stay, and length of reservations. If the travel plan given by a user is over-subscribed, that is, no feasible choices exists that can meet all temporal constraints in the problem, the advisory system suggests trade-offs between destinations and trip durations to

restore the feasibility of the user's plan. The BCDR approach was later extended with a controllability model, Conflict-Directed Relaxation with Uncertainty (BDRU) [114], for solving problems with temporal uncertainty, which are often encountered in real-world scenarios. However, both of these algorithms only work with a static set of choices and objective function defined by the user beforehand.

With Semantic Relaxation, the relaxation algorithm's set of solutions can be extended to include new options generated using the Semantic Memory, ranking candidates with the help of the user's preferences and behavior.

2.1.4.2 Conflict Resolution

The temporal relaxation algorithm uses a conflict-directed search strategy for enumerating alternative solutions. It was first introduced by Conflict-directed A* [112] for hardware diagnosis problems with discrete domain variables. The BCDR algorithm extends it to handle continuous variables and constraints, and later the CDRU algorithm extends the conflict learning and resolution process to account for uncertain durations. Given an uncontrollable temporal problem, CDRU can explain the cause of failure as conflicting sets of choices and constraints, and enumerates preferred continuous relaxations that restore the controllability.

Our integration is implemented based on the CDRU algorithm, which has two major steps:

- Conflict Learning: Given an uncontrollable temporal problem, detect and extract conflicts using a dynamic controllability checking algorithm. This step remains unchanged in our integration: a conflict is defined as a mixed set of constraints and choices that are not dynamically controllable.

- Conflict Resolution: Given a set of conflicts, compute preferred relaxations to their constraints and choices to eliminate these conflicts. We fold in semantic relaxation in this step: in addition to continuously relaxing the bounds of temporal constraints, we will also query the semantic relaxation generator to supply

additional domain assignments. These new domain assignments will deactivate one or more constraints, and hence resolve the conflicts.

2.1.4.3 Candidate Generation Integration

The CDRU algorithm uses the resolutions to unresolved conflicts to expand the search tree. Previously, there were two options for resolving a conflict: alternative assignments and constraint relaxations. In our integration of semantic and temporal relaxations, a third option is added to represent the additional conflict resolutions from the semantic relaxation generator. The three options are attempted in the following order:

- Change assignments to deactivate constraints.
- If no alternative assignments exists for some variables, query the Semantic Memory for additional domain assignments, which can also deactivate constraints and resolve conflicts.
- Finally, compute continuous temporal relaxations by relaxing the temporal bounds of requirement constraints or tightening the temporal bounds of contingent constraints.

The conflict resolution process, implemented in Function EXPANDONCONFLICT (Algorithm 2.1), is separated into two stages. The first stage (Line 3-12) computes discrete resolutions to conflicts using alternative assignments. We look for alternative values in the variable domains that can deactivate one or more constraints in the conflict, and use them to generate new candidates. If no such assignment can be found, we query the semantic relaxation generator to supply additional domain assignments as alternatives to resolve the conflict (Line 7).

The input to the semantic relaxation generator is a pair: the domain assignment to be relaxed (a) and the number of additional options (1 in our implementation). If an alternative can be found, the generator will return it and its utility, which is the likelihood that the user will prefer it.

Algorithm 2.1: Function EXPANDONCONFLICT

Input: A candidate to expand $Cand\langle A, R_S, R_T, C_{cont}\rangle$ and an unresolved conflict $currCFT$.

Output: A set of new candidates $newCands$ that extend $Cand$.

Initialization:

$newCands \leftarrow \{\}$;

$CFTs \leftarrow C_{cont} \cup \{currCFT\}$; conflicts to be resolved continuously;

Algorithm:

for $a \in currCFT$ **do**

  $A_{alter} = A_{alter} \cup$GETALTERNATIVES$(a) \cup$GETALTERNATIVES$(label(a))$;

end

if $A_{alter} == \phi$ **then**

  $A_{alter} = A_{alter} \cup$GETSEMANTICRELAXATION$(a, 1)$
  $\cup$GETSEMANTICRELAXATION$(label(a), 1)$;

end

for $a_{extend} \in A_{alter}$ **do**

  $Cand_{new} \leftarrow \langle A \cup \{a_{extend}\}, R, C_r, C_{cont}\rangle$;

  $newCands \leftarrow newCands \cup Cand_{new}$;

end

$\langle E_r, E_u, N_{value}\rangle \leftarrow$EXTRACTCONSTRAINTS$(CFTs)$;

$f_{obj} \leftarrow \sum_{e \in E_r \cup E_u} f_e(\Delta e)$;

$\langle R_e, R_u\rangle \leftarrow$OPTIMIZE$(f_{obj}, \langle E_r, E_u, N_{value}\rangle)$;

if $R_e \neq null$ *OR* $R_u \neq null$ **then**

  $Cand_{new} \leftarrow \langle A, R_e, R_u, C_r, C_{cont}\rangle$;

  $newCands \leftarrow newCands \cup Cand_{new}$;

end

`return` $newCands$;

For example, for a conflict that involves the assignment *Lunch = Lemongrass*[1], we can resolve the conflict using its alternative in the domain of variable *Lunch*: *Lunch=Wild Ginger*. If none of the these assignments is available, the semantic relaxation generator will compute an additional domain assignment that are similar to the existing ones, such as *Lunch=Lotus Asian*. This assignment will then be used to resolve this conflict and extend the search tree.

Note that during the expansion of the search tree, we need to create an additional search node for each ExpandOnVariable and ExpandOnConflict step, which reserves space for semantic relaxation. For example, while we expand on variable *Lunch*, an

[1] *Lemongrass*, *Wild Ginger*, and *Lotus Asian* are the names of assignments to *Lunch* corresponding to different restaurants in Seattle.

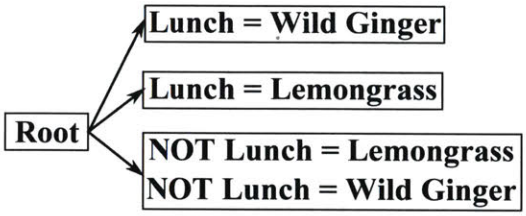

Figure 2-5: Reserving space for semantic relaxation using an additional search node

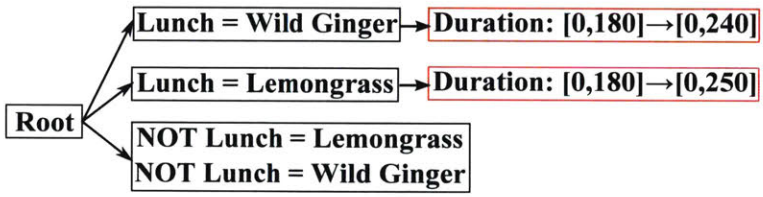

Figure 2-6: Resolving conflict using temporal relaxation

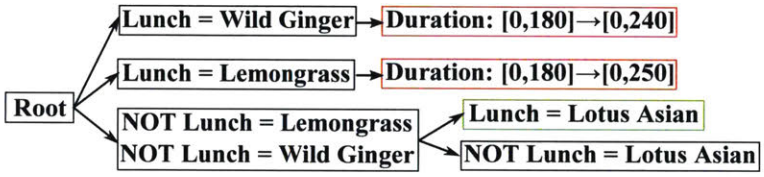

Figure 2-7: Resolving conflict using semantic relaxation

additional search node, $\neg Lunch = Lemongrass, \neg Lunch = WildGinger$, is created (Figure 2-5). When both domain assignments of variable *Lunch* are evaluated to be infeasible (Figure 2-6), we extend this node to create new candidates generated from semantic relaxation (Figure 2-7). Similarly, an additional node of $\neg Lunch = LotusAsian$ is created next to $Lunch = LotusAsian$ in case we need to further relax the domain of *Lunch*.

The second stage (Line 13-19) implements the third conflict resolution option: continuously relaxing temporal constraints. This stage remains unchanged in our integration. The key idea is that if the amount of modification we applied to the constraints in a conflict exceeds the magnitude of its negative cycle's value, this cycle will be eliminated and the conflict is then resolved. This allows us to formulate the problem of computing preferred continuous relaxations as a linear optimization problem. The variables in this problem are the modifications we should apply to the temporal bounds of relaxable constraints. If the LP is feasible and a solution is

returned by the optimizer (Line 15), we use the solution to construct a new candidate. Finally, this candidate and the candidates generated in the first stage will be returned by function EXPANDONCONFLICT to the main CDRU algorithm in [114].

2.2 Experiment

To test that our recommendation system's decisions align with humans, we simulate a trip planning scenario where our system helps plan a trip with a human. To validate the quality of our learnt ranking metric and the effectiveness of our approach, we compare restaurant recommendations for a user made by human subjects with those made by our system, when both are given the same background information about the user.

I will first describe how we construct the Semantic Memory, then describe our experimental protocol when collecting human recommendations, and finally explain our evaluation method.

2.2.1 Semantic Memory

2.2.1.1 Dataset

In our experiment, we learn an ontology similar to the one shown in (**Figure 2-4**) by collecting reviews and metadata for all restaurants in Seattle listed on Yelp in December 2014. After collection and removal of duplicates we obtain information about 6122 restaurant documents. We convert all metadata to a binary feature vector with 198 dimensions. This feature vector has 188 independent category features, and 2 sets of 5 mutually exclusive labels: pricing: *0$ to 4$*, and rating: *1 to 5 stars.*

2.2.1.2 Reader Module

We learn item embeddings for each restaurant by predicting the associated metadata using the Reader Module shown in (**Figure 2-1**), with the restaurant's item vector and a random word window from the associated review text. Through cross validation

on held out word windows we find that a word and item embedding dimension of 20 works best. We train our network using stochastic gradient descent with the AdaGrad update rule [22], until prediction error for metadata using the held-out word windows stops decreasing.

2.2.1.3 Hierarchical Clustering

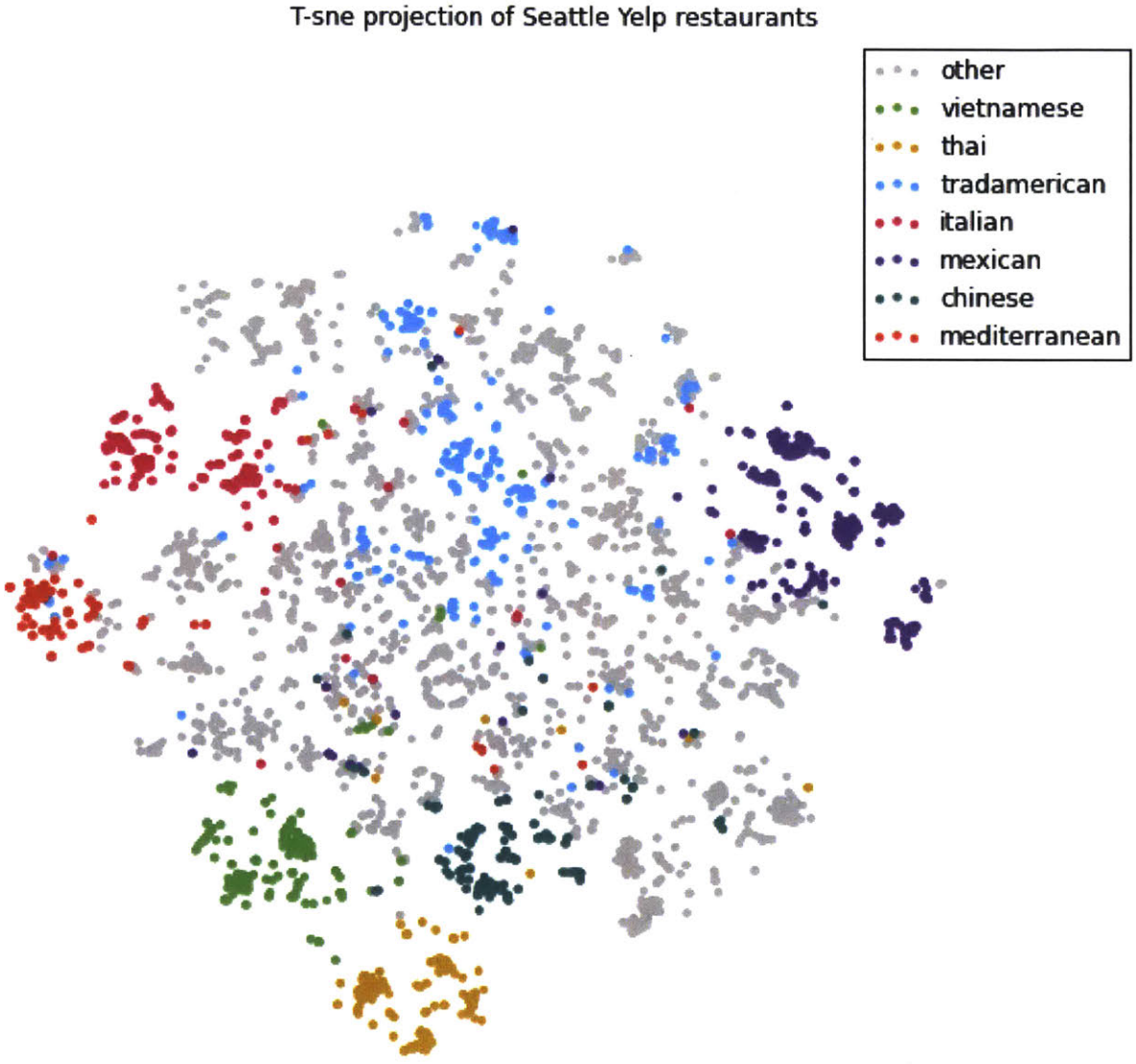

Figure 2-8: 2D projection obtained via T-SNE[99] of the distributed representation for all Seattle restaurants present in Yelp. The clusters in the figure correspond to places with semantically similar cuisines or with overlapping vocabulary in their reviews.

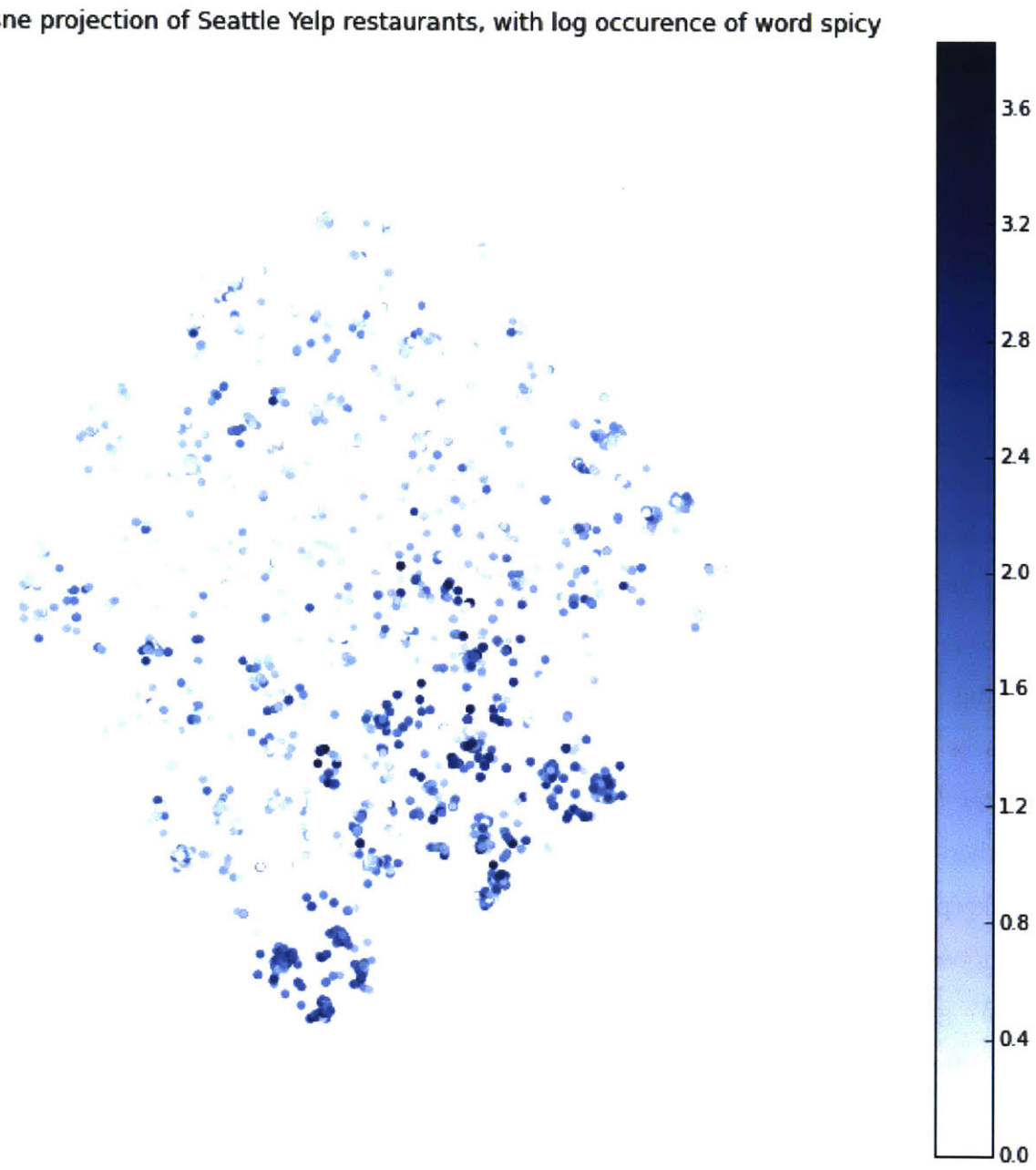

Figure 2-9: A second 2D projection obtained via T-SNE[99] of the distributed representation for all Seattle restaurants present in Yelp. In this figure, the intensity of the color of the dots corresponds to the log occurrence of the word "spicy". The bottom half of the learnt embedding space correlates with the presence of spicy food, a feature only made available through review text.

The learn item vectors for each restaurant visible in (**Figure 2-8**) and (**Figure 2-9**) are then clustered using average hierarchical clustering [88, 58]. We use the resulting binary tree to find the Laplacian, Δ, and the covariance matrix K for inference and

ranking. A real example output of this clustering algorithm using a small subset of our data is visible in (**Figure 2-4**).

2.2.2 Experimental Protocol

We conduct our experiment with human subjects on Mechanical Turk by simulating a recommendation situation with past user behavior. Each worker is asked to recommend one of 3 restaurants in a list after reading a description of the restaurant and viewing the same metadata as our system. The worker also receives a description of the places liked by the person they are a recommending a restaurant to. Additionally, we ask the workers to select the rationale for their decision among *cuisine, rating, price,* or *style.*

To generate recommendation instances we simulate a user using simple rules concerning *rating, price, categories,* or *value=rating−price.* Using those rules we select 4 places at random which all match a particular rule's criteria (e.g. *must have rating greater than 3 stars*, *category must be Italian*, etc..), and 2 that do not fit the rule. We then present 3 of the rule-conforming restaurants as part of the user's description, and keep the remaining passing and 2 rule-breaking restaurants as options for workers to choose from.

We generate 108 unique instances using 9 different rules, and show each instance to 5 different workers.

2.2.3 Evaluation

We evaluate the performance of our approach by comparing the agreement between the majority of human responses and the recommendations made using different learning algorithms. We compare our approach to several baselines, along with Paragraph Vector Distributed Memory [57], a state of the art information retrieval and semantic hashing model, and probabilistic-LSA [37], a powerful topic modelling technique for obtaining summary vectors from documents We use as our baseline Logistic Regression and RBF-kernel SVMs [18] with the top 10,000 dimensions from Term-Frequency

Inverse Document Frequency (TF-IDF) [81] vectors as inputs. Using the learnt embeddings and representations for items from each technique we construct different "Semantic Memories" for each, and rank each recommendation instance's options, and pick the top one.

2.3 Results

Table 2.1: Comparison of model agreement with majority of human judges.

	% time model agrees with majority				
Learning Method	Overall	Price	Cuisine	Rating	Value
Logistic Reg. w/. TF-IDF	33.33	41.67	31.94	25.00	41.67
SVM w/. TF-IDF	25.00	25.00	26.39	8.33	33.33
P-LSA [37]	31.48	41.67	33.33	33.33	8.33
PVDM [57]	36.11	16.67	38.89	25.00	**50.0**
Object LM	40.74	33.33	43.10	25.00	**50.0**
Object LM + Quadratic	**43.52**	**50.0**	**44.44**	**33.33**	41.67

The percent agreement between the majority of human's recommendations and the compared approaches is shown in (**Table 2.1**). Results for baselines and alternate learning algorithms are in the upper half of the table. Results for the proposed learning method are shown in the lower half of the table. In the columns of the table we show agreement results for all instances, and recommendation instances that were constructed using specific rules.

In all cases we find that our approach outperforms or matches existing techniques when comparing alignment with human judgment.

2.4 Analysis

The use of a richer representation was not only beneficial from a user interaction point of view, but also improved performance of the system by allowing the combination of explicit and implicit features.

2.4.1 Semantic Memory

In our experiment we find that a combination of explicit and implicit features improves agreement with human judgment by significant amount (6.4% overall) (**Table 2.1**). Moreover, using this technique we were able to construct a representation that can provide a rationale for a machine's recommendation by using a human understandable metric. In particular, we find that information only available from the implicit features in the review text is reflected in the learnt item topology: spicy food is not one of the explicit categories, yet a cluster of "spicy" places is found (**Figure 2-9**) bridging Mexican and asian cuisine clusters in (**Figure 2-8**).

2.4.2 Semantic Relaxation

In our experiment with Semantic Relaxation, with the modification to CDRU that uses a Ontological User Profile to suggest alternative solutions, we open the possibility for more work on making the objective function of constraint relaxation algorithms dynamic.

Today, the Semantic Relaxation algorithm serves only as a candidate generator, however, an exciting area of research would be to learn how to richly couple the original objective with semantic constraints so that richer and more complex queries can be made by the user and enforced by the planner.

Chapter 3

Sentiment Analysis

Human language is indissociable from emotion. In order for a computer to understand the way we speak or communicate it must have an ability to comprehend or appreciate the underlying tone. Depending on context "Okay" may signify agreement, resignation, or simply acknowledgment. Understanding the differences between these cases is crucial for a digital personal assistant to interact in a meaningful way with humans.

In this chapter, I will present a novel machine learning framework, Memory Backpack (MB), that establishes a new state of the art results on the *Stanford Sentiment Treebank*, a challenging Sentiment Analysis benchmark task. Moreover, unlike competing approaches, the MB gives a visual explanation for its predictions.

3.1 Approach

My proposed framework, MB, is a neural network that makes iterative changes to its prediction. A penalty is added to the objective function for each update, encouraging the network to commit and only make changes when new information is present. In Chapter 4 of this thesis, I provide a more in-depth analysis and discussion of this penalty function, Occam's Gates, which encourages sparse activation of gates and favours an Occam's Razor solution to emerge during training.

This framework resembles the work of [87] where a quadratic form controls the

flow of information between different nodes of a tree, and the *episodic memory* in [55], where a network uses gates to update its states, before making a prediction.

3.1.1 Network Architecture

The network receives as input a sequence of words or vectors $S = [s_1, \ldots, s_n]$ and must output a probability distribution or target state t_{final}. As the sequence is read the network produces intermediate states $t_1, \ldots, t_{\text{final}}$. The network is defined by two key components:

1. a **processing module**, which takes the previous network state and current input $\{t_{i-1}, s_i\}$, and forms a candidate prediction $\tilde{t}_i$,
2. a **memory gate**, that takes the previous prediction and current input $\{t_{i-1}, s_i\}$, and the candidate state $\tilde{t}_i$, and combines them into a new state t_i as a linear combination of the candidate and past state using a gating scalar value $g_i \in [0, 1]$):

$$t_i = \tilde{t}_{i-1} \odot g_i + (1 - g_i) \odot t_{i-1}.$$

This framework has two key features:

1. The *memory gate* captures climactic moments as information sources.
2. Activation of *input* and *memory gates* act as highlighters, which indicate the network's attention, and gives clues about the location of new information in the source sequence.

3.1.2 Sentiment Analysis Network Architecture

The specific module instantiation when performing Sentiment Analysis is as follows:

3.1.2.1 Processing Module

For Sentiment Analysis the **processing module** used for making predictions is a bidirectional LSTM [27], which allows each step of the prediction to make use of the

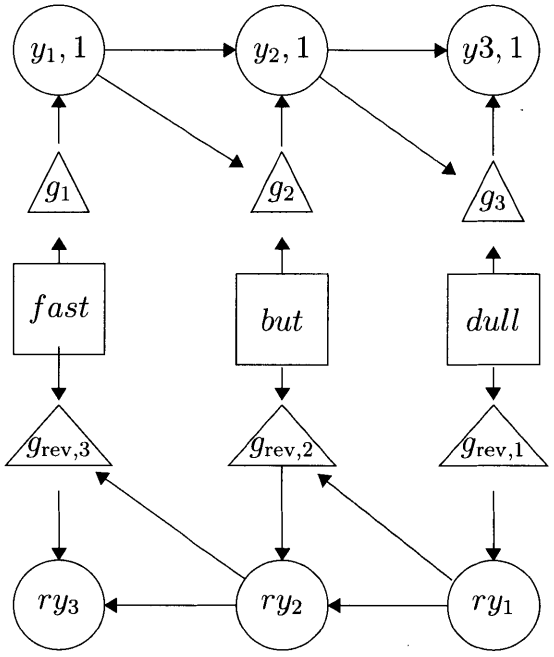

Figure 3-1: A bidirectional LSTM that uses input gates to control what subset of the input is passed on for later processing.

sentence context in both directions (**Figure 3-1**).

At each step of reading the forward and backward states of the bidirectional LSTM are fed to a *reader LSTM* that fuses the observations. The hidden state of this *reader LSTM* is projected using an affine map into a 5 class probability distribution normalised using exponential normalization (Softmax). The network diagram is shown in (**Figure 3-2**) with input gates between words and LSTM states $\{y_{1,2,3}, ry_{1,2,3}\}$ omitted for clarity.

3.1.2.2 Memory Gate

Predictions at every timestep are linearly combined with past predictions via a memory gate. The memory gate's purpose is to control how much of the new prediction should replace the previous prediction by outputting an interpolation value between 0 and 1, with 1 signifying that the previous prediction should be entirely replaced. The specific equations for the memory gate are identical to those provided earlier (**Section 4.1.1**). In this instance, a quadratic gating function (**Section 4.3**) was

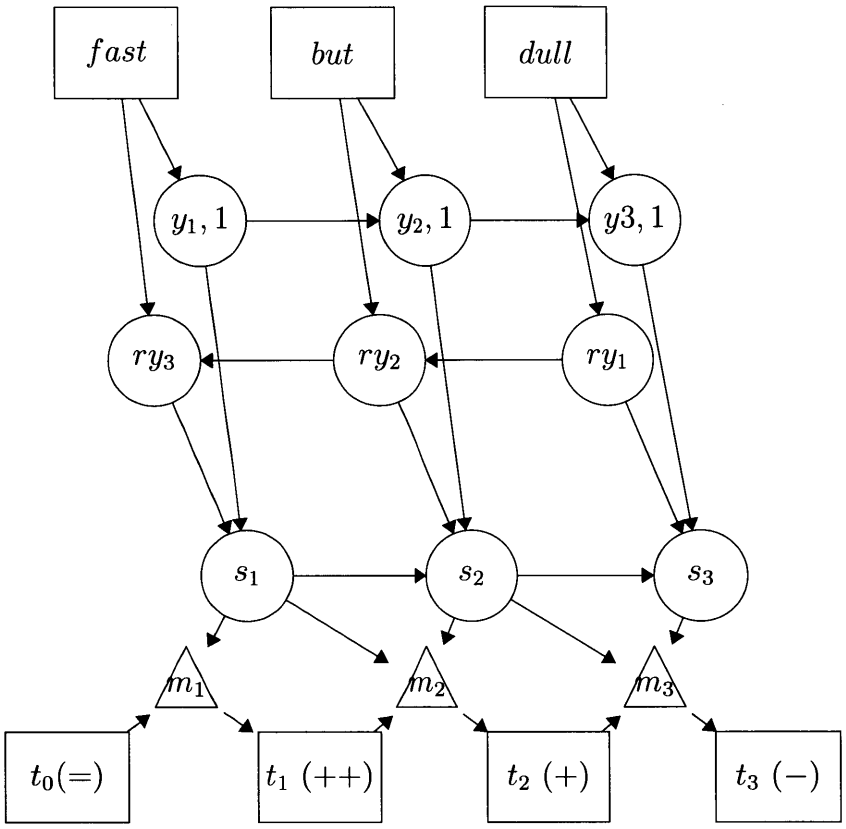

Figure 3-2: The sentence "fast but dull" is passed to a bidirectional LSTM. The forward and backward states of the network are processed by a left to right reader LSTM ($s_{1,2,3}$). The reader produces predictions $\tilde{t}_{1,2,3}$ which are gated and fused with the previous predictions $t_{0,1,2}$.

chosen after selection via hyper parameter search on a held-out validation dataset.

3.1.3 Vocabulary Expansion

During evaluation of the model over new data, previously unseen words do not have a vector representation. The specific way this is handled becomes important when considering real world text where slang, misspellings, or typos, introduce many missing words. For example, in the held out validation set of the *Stanford Sentiment Treebank*, 61.3% of sentences contain at least one word that does not appear in the training set. In my experiments, I compared two different strategies for coping with these cases, described below.

3.1.3.1 Unknown Word Symbol

Unseen words are replaced with a special unknown word symbol, <UNK> [16]. To learn a vector representation for <UNK>, rare words in the training data are replaced in the sentences by <UNK> with some chosen probability p. During training, this approach allows <UNK>'s word vector parameters to be learnt. During evaluation, unseen words are substituted by <UNK> and inherit a proxy embedding.

3.1.3.2 Expansion

Another strategy is to learn a mapping, $f : E_{\text{pre-trained}} \rightarrow E_{\text{backpack}}$, between a larger set of pre-trained word vectors, $E_{\text{pre-trained}}$, and those ones trained by MB, E_{backpack}.

This mapping would allow me to use pre-trained word vectors for a missing word w_{missing}, $\text{Embed}(w_{\text{missing}}, E_{\text{pre-trained}}) = \vec{w}_{\text{missing}}$, and convert them into their equivalent in the trained embedding space:

$$f(\vec{w}_{\text{missing}}) = \vec{w}_{\text{missing,converted}}.$$

To learn this mapping, I follow the approach taken [52, 64], where they find a linear mapping between word spaces parametrised by a matrix, $\mathbf{W}_{\text{translator}}$, and an offset, $\vec{b}_{\text{translator}}$, such that:

$$\vec{v}' = \mathbf{W}_{\text{translator}} \cdot \vec{v} + \vec{b}_{\text{translator}},$$

where $\vec{v}' \in E_{\text{backpack}}$ and $\vec{v} \in E_{\text{pre-trained}}$.

3.2 Experiment

In this thesis, I compared my approach with others on a benchmark sentiment analysis task, the *Stanford Sentiment Treebank* (SST) [87]. This dataset is a collection of 11,855 sentences extracted from movie reviews that specifically tests the effects of negation, emphasis, and rare words. Each sentence has a sentiment annotations

from 5 classes: {*terrible, bad, neutral, good, terrific*}, for the 215,154 unique sub-phrases obtained after parsing each sentence using the Stanford Parser. My results are compared with the current state of the art [94, 55], along with two baseline architectures: a bidirectional LSTM, and a unidirectional LSTM. The model and its baselines were implemented using **Dali**[1], an open-source deep learning framework.

3.2.1 Training

Training is performed using the provided *train-validation-test* split of the dataset. Because the MB does not make use of syntactic parse trees, the labeled trees are converted to phrases during training using the same strategy as [94][2]: the full sentence along with a random sample of labeled spans is extracted from the original tree, and each is treated as a separate training example.

Specific model hyper parameters are selected using grid search with the validation set. The best configuration found uses pre-trained 300 dimensional word vectors from Glove [74], LSTMs with 150 memory cells, and gradient descent using the AdaGrad update rule with a learning rate of 0.05, and an L_2 weight regularization of 0.0001[3]. Early stopping is performed when accuracy stops increasing on the validation set.

3.2.2 Evaluation

The model is evaluated by looking at the prediction accuracy for the label of the root of the trees in the test set. Two separate evaluations are performed: 5-class (a.k.a. fine-grained accuracy): {*terrible, bad, neutral, good, terrific*}, and binary accuracy: { *negative, positive*}[4].

[1] `https://github.com/JonathanRaiman/Dali`

[2] `https://github.com/stanfordnlp/treelstm/`

[3] Interestingly the best hyperparemeters for MB are the same as those used by [94] in their baseline architectures.

[4] Binary accuracy is measured by grouping all negative sentences and positive sentences, and skipping neutral sentences.

3.3 Results

In this section, I present results comparing the MB with state of art approaches, and also provide examples for the visual explanations provided by MB over unseen sentences.

3.3.1 Comparison

Table 3.1: Comparison of Root Accuracy on the *Stanford Sentiment Treebank*

Model	Binary	5-class
LSTM (mine)	83.0%	43.78%
MV-RNN [86]	82.9%	44.4%
Bidirectional LSTM (mine)	83.4%	44.5%
RNTN [87]	85.4%	45.7%
DMN [55]	88.3%	50.3%
Tree-LSTM [94]	86.9%	50.6%
Memory Backpack		
MB + mapping	89.07	51.45%
MB	**89.13%**	51.53%
Memory Backpack Ensemble		
3×MB + mapping	89.02	51.86%
3×MB	89.02	**51.90%**

Results for root accuracy using the LSTM and bidirectional LSTM, along with state of the art results can be found in the upper portion of (**Table 3.1**), with MB results shown in the lower two tables. The bottom table contains results obtained by averaging the predictions from 3 MB models initialized with different random seeds.

The MB outperforms the bidirectional LSTM baseline by 5.73% on binary accuracy, and 7.03% on 5-class accuracy.

The MB attains 51.9% fine-grained root accuracy and 89.13% binary accuracy, establishing a new state of the art for Sentiment Analysis on the Stanford Sentiment Treebank. The MB outperforms to the best of my knowledge all other approaches, including the current 5-class state of the art [94], and the current binary state of the art [55].

As visible in the table, the addition of a mapping for vocabulary expansion slightly reduces the accuracy for binary and fine-grained accuracy in the single model case, and has no negative impact on binary accuracy for the ensemble case.

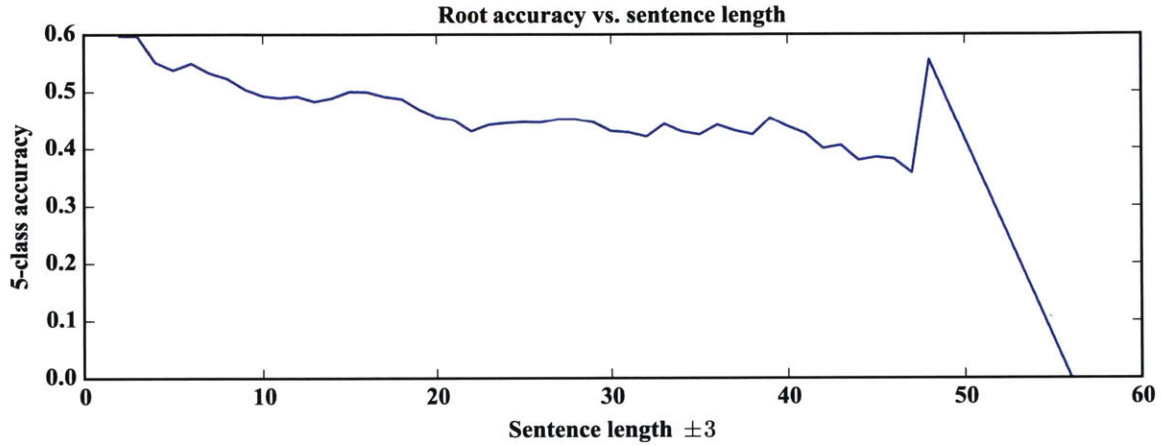

Figure 3-3: Root accuracy initially drops as a function of sentence length, and steadies around 45%.

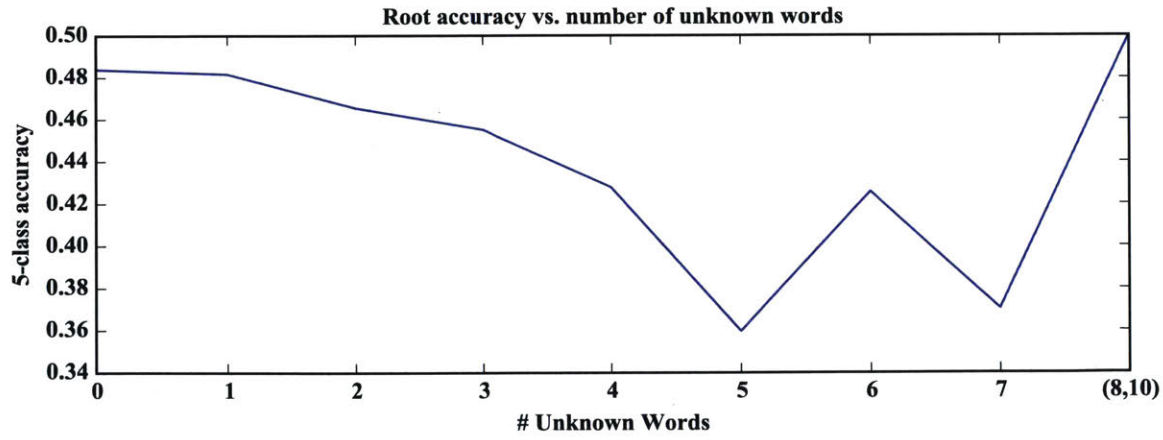

Figure 3-4: Root accuracy drops dramatically with the number of unknown words in a sentence (apart from outliers at 8 and 10).

We can additionally observe the effect of unknown words and sentence length on the MB's performance in (**Figure 3-3**) and (**Figure 3-4**). Longer sentences reduce the accuracy of the system, while unknown words have a more severe impact.

3.3.2 Visual Explanation

The MB differs from any of its counterparts in its ability to provide an explanation for the prediction performed by the machine.

3.3.2.1 Sentiment Reversal Visual

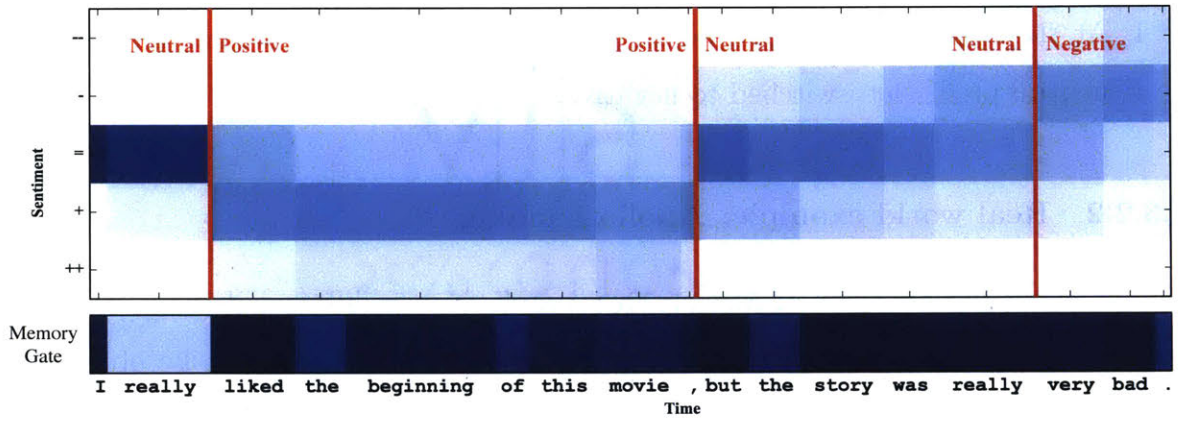

Figure 3-5: An example of MB switching emotional state from positive to negative with sentence progression

Let us take as an example a sentence from the context of movie reviews: "I really liked the beginning of this movie, but the story was really very bad.", with the output of MB shown in (**Figure 3-5**). In the figure, the first 5 rows indicate the probability under MB of the sentence at this point of the reading belonging to a particular sentiment class (e.g. row 1 corresponds to the probability of being *terrible*/−−, and row 5 corresponds to *terrific*/++). The bottom row is the interpolation value m_i shown earlier in (**Figure 3-2**). This bottom row corresponds to the intensity of the update to the MB's memory (a value of 1 completely replaces the current prediction, while 0 keeps its unchanged).

In this example, the gate (lowest row) is at its peak at 4 points: "I", "liked", "but", "bad." At those points in the reading of the story the trained MB model picks up on changes in the emotional state of the sentence:

1. At "I" the sentence has just begun, the model switches from a uniform distribution to one with its maximum on neutral.

2. At "liked" the first positive word in the sentence is picked up.

3. At "but" the initially positive state of the system (still present at ",") switches to neutral, reflecting the reversion.

4. At "bad" the negative information is incorporated into the prediction, and the neutral prediction switched to negative.

3.3.2.2 Real world example: Apollo Landing

By observing the behavior of MB on an example from a different domain [8] than its training data, a better picture of the strength and weaknesses can be obtained. Words between two asterisks `*WORD*` were unknown to MB and projected using the method described in (3.1.3.2).

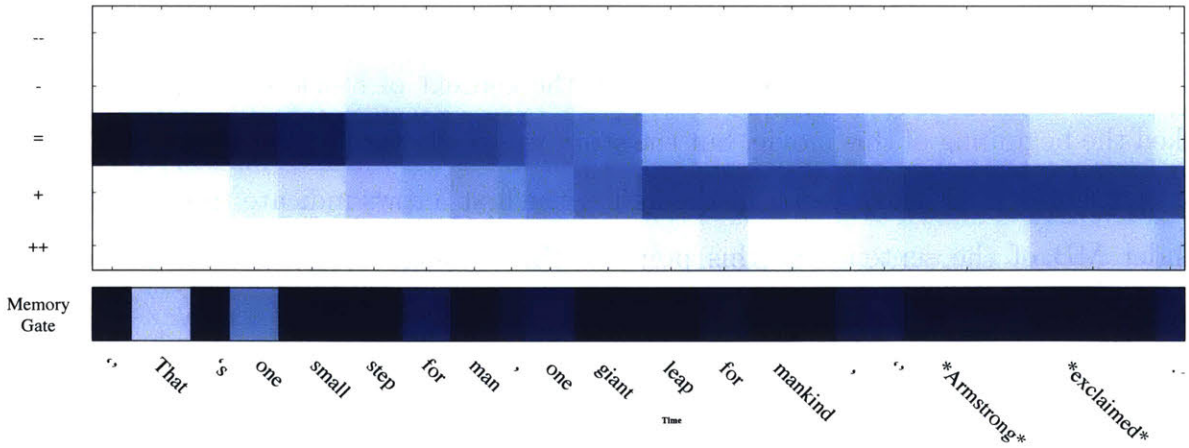

Figure 3-6: Sentiment Analysis over text in a separate domain. Words between asterisks were not seen during training, yet the network still reacts appropriately.

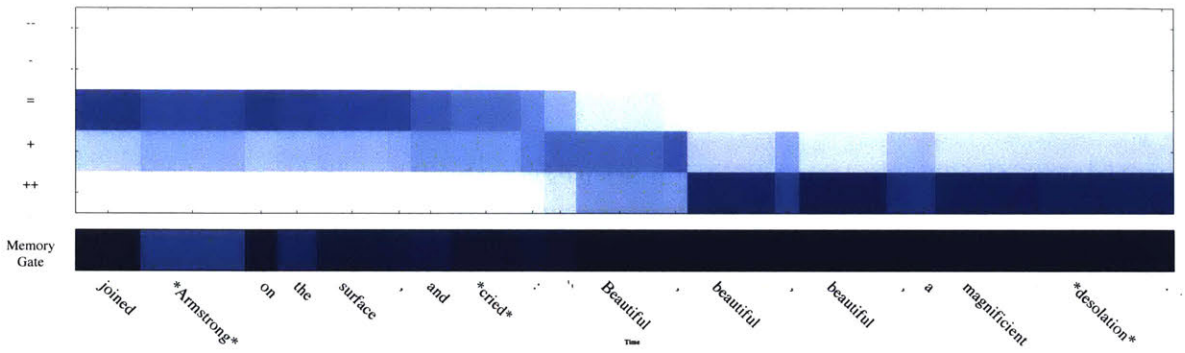

Figure 3-7: Sentiment Analysis over text in a separate domain. The words "desolation" and "Armstrong" were not seen during training, yet the name is kept neutral and desolation is detected as being negative

In the examples shown above, areas that are sentiment-rich also correlate with areas of tension and celebration in this scientific achievement.

3.4 Analysis

The Memory Backpack is a general machine learning framework for processing sequences. It is unique in its ability to provide control and visualization of the flow of information to the prediction. On a challenging Sentiment Analysis dataset, a single trained model establishes a new state of the art accuracy for binary and fine-grained accuracy without any linguistical data or parse trees, while also providing for every example a visual explanation supporting the prediction made.

In this experiment several conclusions and observations can be drawn concerning the importance of a memory gate, vocabulary expansion, and the importance of an interpretable output.

3.4.1 Memory Gate

In my experiments I find that the use of a memory gate allowing the current prediction of the system to be part of the state improves the performance of RNNs by 5.73% on binary accuracy, and 7.03% on 5-class accuracy. It supports the claim that the ability to explicitly model and capture changes in the prediction improves accuracy.

3.4.2 Vocabulary Expansion

Surprisingly, the vocabulary expansion strategy rather than improving performance over unseen words, instead has a slightly negative impact on prediction accuracy(-0.04% on fine-grained accuracy, and -0.05% on binary accuracy) (**Table 3.1**). One possible conclusion from this is that pre-trained word vectors found in Glove [74] already contain sufficient information for Sentiment Analysis. Secondly, it appears that the new location of word vectors after training does not provide sufficient information to beneficially project non re-trained vectors and construct an informative mapping. I leave further investigation into which projection method might work best as future work.

3.4.3 Interpretability

The MB is unique in its representation of internal state that remains interpretable by humans all along processing. As visible in (**Figure 3-5**), as a sequence of words is read, the emotional memory of the network is far from constant: on the contrary, changes in tone or direction of the text are reflected at each timestep.

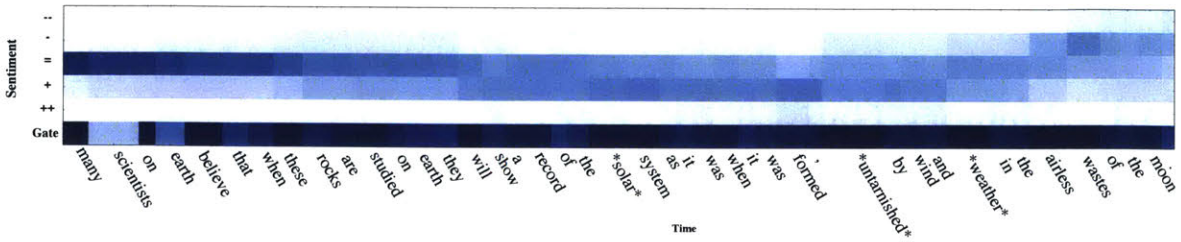

Figure 3-8: Sentiment Analysis over text in a separate domain. The long sentence and lack of clear sentiment markers causes indecision in the network reflected by the oscillation around neutral.

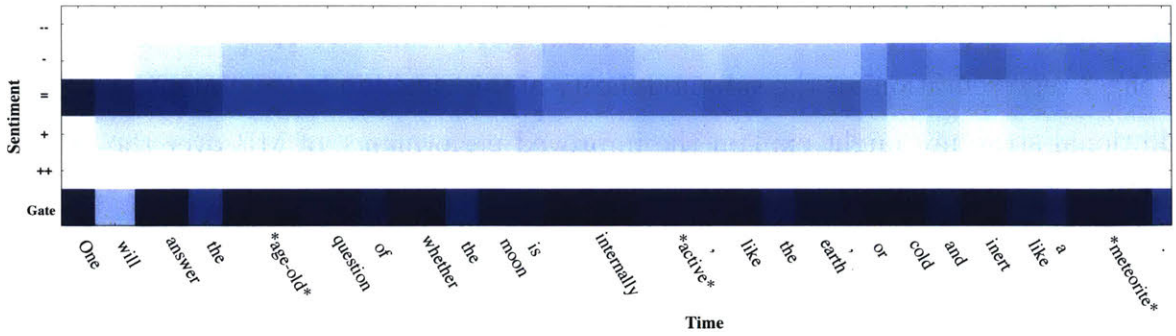

Figure 3-9: Sentiment Analysis over text in a separate domain. Words outside of the training data, which carry different meaning for movie reviews, like "cold" cause the network to view the sentence as negative.

It is also important to understand the failure modes of the system more directly. A weakness of MB is visible in longer and more neutral sentences such as (**Figure 3-8**) and (**Figure 3-9**), where the predicted distribution is undecided, causing oscillation and unpredictable behavior from the gates.

In order to achieve true Man-Machine interaction, it is crucial that the machine's behavior is visible to the human to create trust and collaboration through a deeper understanding.

3.4.4 Comparison

The nature of the memory updates is a fundamental difference between the DMN and the MB: in my framework, memory is used to store the current "emotional state" of the system, and is thus relatable to the output and interpretable by a human at every word read from the sentence. Memory in the DMN stores recurrent neural network state snapshots, which are not immediately relatable to sentiment predictions[5]. Moreover, the DMN's reliance on multiple reads of a sentence disconnects memory updates from specific moments in the sentence. While the memory representation in the MB helps with interpretation, it constrains updates to the prediction to only be sentiment label probability distributions, thus long term state is only stored using the hidden units of LSTM cells. This representation, modelling the state of the

[5]These memories, however, serve at a later stage of the computation to output a sentiment label in plain english using a separate decoder RNN.

Sentiment Analysis system as directly containing predictions, appears to provide a stronger representation for the sub modularity of this signal in human language. This additional structure might explain the improved performance of MB over the DMN and Tree-LSTM (**Table 3.1**).

3.4.5 Conclusion

In this chapter, I introduced Memory Backpack, a machine learning framework that allows visualization and interpretation of a recurrent neural network through the use of sparsity and gating mechanisms, and achieves a new state of the art result.

The underlying framework is general and can be applied to other sequence prediction problems where suddens changes and events in the sequence are the source of information.

Chapter 4

Occam's Gates

> Pluralitas non est ponenda sine necessitate.
> *Plurality must never be posited without necessity.*
>
> *Guilelmus de Ockham,*
> *Summa totius Logicae (1495)[20]*

The ability to generalize from experience is a fundamental part of an artificial intelligence. In order to incorporate and react to new information about the world, the machine must be able to extract reusable lessons from experience. In this chapter I present a complimentary objective function for training Recurrent Neural Networks (RNNs) inspired by Guilelmus de Ockham's saying, "Plurality must never be posited without necessity" [20]. This objective function allows machines, which process sequential data using RNNs, to explain their decision process by pinpointing what temporal regions of an input sequence were attended to.

The intuition behind Ockham's saying could be translated into considering that only a subset of the information in a problem is necessary and sufficient to respond. In a sequence learning problem this would mean that only a portion of the time steps carry useful information. Practically, an RNN can possess input gates that include or ignore a timestep's entire input. The behavior of these gates can be coerced to be sparse by enriching the original objective function for the sequence learning problem

in the following way: the sum of the activations of the gating units g_i weighed by a hyper-parameter λ_{sparse} is added to the original objective J:

$$J^* = J + \lambda_{\text{sparse}} \cdot \sum_i g_i. \tag{4.1}$$

This objective function change encourages the network to focus on a subset of the inputs during inference, and thus generalize better to unseen sequences by forcing insensitivity to minor changes in the input. I demonstrate the effectiveness of this approach on three separate sequence learning tasks: sentiment analysis, paraphrase recognition, and question answering.

4.1 Approach

In order to give RNNs added transparency in their decision process it is possible to add gates to their inputs at every timestep, and apply Occam's Razor over our training data to find in each example a minimal set of useful inputs over time. To achieve this property, I enforce sparsity in the activation of the RNNs' input gates. In a sentiment classification problem for instance, gates would ideally only fire for emotionally loaded words, and stay dormant otherwise.

Because Occam's Gates relies on the addition of input gates to RNNs, I make the assumption that the vector input at each time-step is an inseparable information unit, like a word, image, or fact. If this assumption holds, then the network is forced to reduce its gate usage through a penalty on the sum of the gate activations, and a solution is found in a local optima where gates are less often active, reducing the sensitivity to minor input variations and highlighting the location of key inputs to the human actor.

I formalise my approach by describing how we enforce sparsity on the gate activations for Long-Short Term Memory networks (LSTM) [36], then I describe how they are used for the different tasks considered in this chapter, and finally I explain the

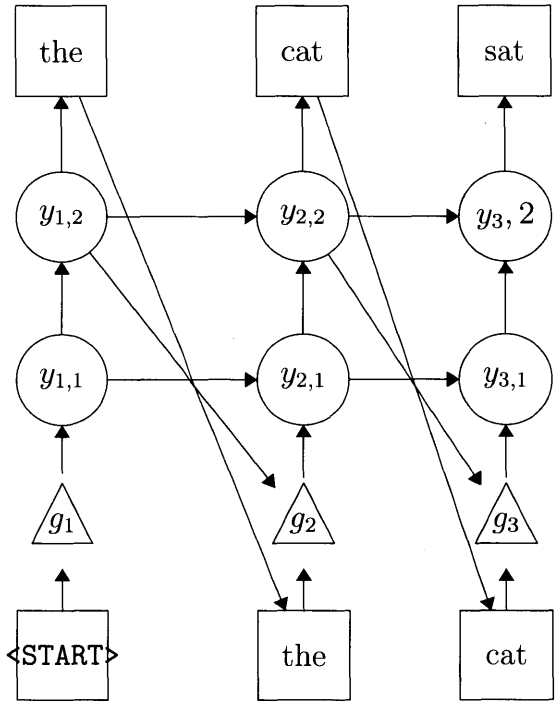

Figure 4-1: An RNN with gates g_1, g_2, g_3 controlling the intensity of the inputs.

sparsity-enforcing objective function and the different annealing regimens considered during training.

4.1.1 Gated LSTMs

In my work I make extensive use of LSTMs, a popular RNN architecture specifically designed to capture long range dependencies and alleviate training difficulties [72, 35]. Since their introduction in 1995, many variants have been proposed [28, 67, 26, 94, 23], however for the purposes of this work I used the *vanilla* version from [28].

While LSTMs are capable of selectively remembering or forget parts of their memory and input, they lack the ability to transform uniformly their input. I extend LSTMs to include an additional gate, g_{occam}, that uniformly multiplies all the inputs simultaneously. The gates are represented using triangles in (**Figure 4-1**), with LSTM states, $y_{i,j}$, represented using circles, recurrent connections shown with horizontal arrows, and input connections shown with upward arrows. The equations for the Gated-LSTM are presented in Table 4.1, with the differences with the regular

Table 4.1: LSTM and Gated LSTM equations. Modifications shown in red.

Description	Symbol	LSTM	Gated LSTM
Occam's gate	g_{occam}	absent	$f_{\text{gate}}(\vec{x}_t, \vec{h}_{t-1})$
gated input	$\vec{x}'_t$	absent	$\vec{x}_t \cdot g_{\text{occam}}$
block input	$\vec{z}_t$	$\tanh\left(\mathbf{W}_z\vec{x}_t + \mathbf{R}_z\vec{y}_{t-1} + \vec{b}_z\right)$	$\tanh\left(\mathbf{W}_z\vec{x}'_t + \mathbf{R}_z\vec{y}_{t-1} + \vec{b}_z\right)$
input gate	$\vec{i}_t$	$\sigma\left(\mathbf{W}_i\vec{x}_t + \mathbf{R}_i\vec{y}_{t-1} + \vec{b}_i\right)$	$\sigma\left(\mathbf{W}_i\vec{x}'_t + \mathbf{R}_i\vec{y}_{t-1} + \vec{b}_i\right)$
forget gate	$\vec{f}_t$	$\sigma\left(\mathbf{W}_f\vec{x}_t + \mathbf{R}_f\vec{y}_{t-1} + \vec{b}_f\right)$	$\sigma\left(\mathbf{W}_f\vec{x}'_t + \mathbf{R}_f\vec{y}_{t-1} + \vec{b}_f\right)$
memory state	$\vec{m}_t$	$\vec{i}_t \odot \vec{z}_t + \vec{f}_t \odot \vec{m}_{t-1}$	identical
output gate	$\vec{o}_t$	$\sigma\left(\mathbf{W}_o\vec{x}_t + \mathbf{R}_o\vec{y}_{t-1} + \vec{b}_o\right)$	$\sigma\left(\mathbf{W}_o\vec{x}'_t + \mathbf{R}_o\vec{y}_{t-1} + \vec{b}_o\right)$
hidden state	$\vec{y}_t$	$\vec{o}_t \odot \tanh\left(\vec{c}_t\right)$	identical

LSTM highlighted in red, with $\sigma(\cdot)$ for the logistic sigmoid function, $\mathbf{W}_{i,z,f,o}$ and $\mathbf{R}_{i,z,f,o}$ for matrices, and $\vec{b}_{z,i,f,o}$ for vectors.

The gating functions $f_{\text{gate}}(\cdot)$ can take multiple forms depending on what order of interaction between state and input is needed to control information flow. I consider two gate functions in this thesis: a linear function of the input $\vec{x}_t$ and a second order gate capable of capturing higher-order interaction. These gating functions are computed as follows, with $\vec{x}_t$ the input to be gated at time t, and $\vec{h}_{t-1}$ the state of the network at time $t-1$:

$$f_{\text{linear}}(\vec{x}_t, \vec{h}_{t-1}) = \sigma(\vec{p}^T \cdot \vec{x}_t + \vec{q}^T \cdot \vec{h}_{t-1} + b), \tag{4.2}$$

$$f_{\text{quad}}(\vec{x}_t, \vec{h}_{t-1}) = \sigma(\vec{h}^T \cdot \mathbf{W} \cdot \vec{x}^T + \vec{p}^T \cdot \vec{x}_t + \vec{q}^T \cdot \vec{h}_{t-1} + b). \tag{4.3}$$

The gates are parametrized by $\vec{p}, \vec{q}$ and b when using $f_{\text{linear}}(\vec{x}_t, \vec{h}_{t-1})$, and $\mathbf{W}, \vec{p}, \vec{q}$ and b when using $f_{\text{quad}}(\vec{x}_t, \vec{h}_{t-1})$.

4.1.1.1 Encoder-Decoder

Occam's Gates can also be applied to encoder-decoder tasks, a special sequence prediction problem where the output of the machine is another sequence. In this sub-

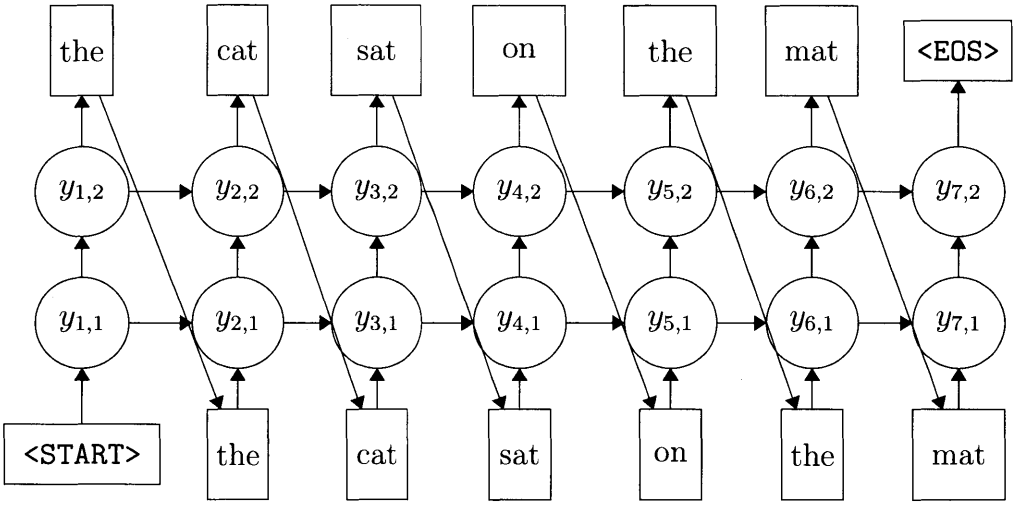

Figure 4-2: English sentence sequence forecasting using Recurrent Neural Networks

section I provide some background on the underlying representation required for this capability, and how this can be applied for the Question Answering and Fact Reading task I validate my approach on.

4.1.1.1.1 Background

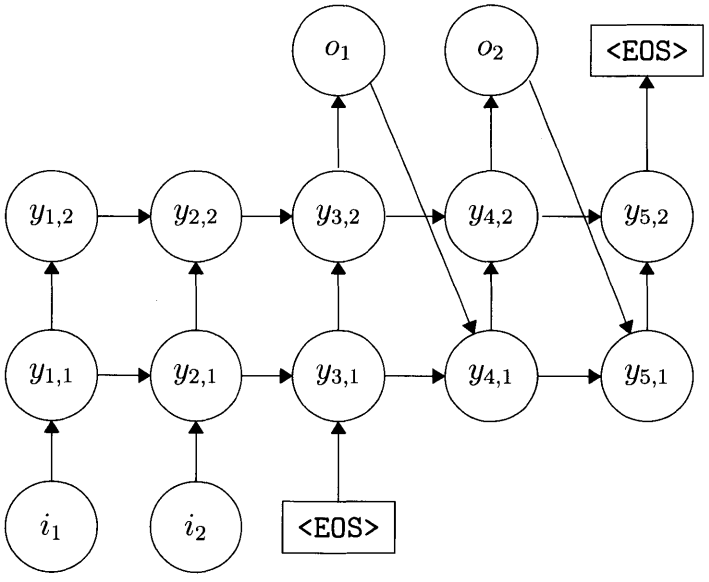

Figure 4-3: A Sequence to Sequence Transducer

The elaboration of more complex RNNs opened the way towards models that could

encode a sequence and decode it into something different. The RNN is taught to wait for special boundary symbols before making predictions and output a special *end of sequence* symbol when the output sequence is over. With these added representational tools, RNNs have been trained to translate between languages [4, 55, 93, 61, 44], produce parse trees [102, 102, 101, 23], answer questions [55, 76], or generate movie dialogues [84, 103]. In the example shown in (**Figure 4-2**) the encode decoding task is simply to predict the next word in the sequence, by adding special boundary symbols: `<START>` to indicate the beginning of a phrase, `<EOS>` to indicate the end of a phrase. The introduction of these boundary symbols has transformed these systems from sequence classifiers, into sequence to sequence transducers (**Figure 4-3**).

4.1.1.1.2 Question Answering with RNNs

To train an RNN to answer questions, the network is taught to produce arbitrary length sequences followed by an end symbol in response to specific question sequences. In this chapter, I train a network to process a question sequence, read several fact sentences, and finally produce a distributed vectorial representation of an answer which is then given to a decoder RNN. The entire system is trained using back-propagation by penalizing the Kullback-Leibler divergence between the expected output token at timestep i ($\{o_1, o_2\}$ in (**Figure 4-3**)) with the correct sequence, with the encoder RNN ($\{y_{1,1}, y_{1,2}, y_{2,1}, y_{2,2}, y_{3,1}, y_{3,2}\}$ in (**Figure 4-3**)) trained by propagating the errors from the decoder RNN backwards in time.

4.1.2 Training Regimens

The ultimate goal of this approach is to use sparsity to highlight the decision process of the machine, increase robustness to minor changes in the input, while maintaining the original power of the network. Forcing sparsity too soon can do more harm than good: a greedy and locally optimal solution is forcing all gates to be closed. To prevent this from happening early exploration is encouraged by progressively increasing the sparsity penalty, λ_{sparse}. I investigate 2 different annealing regimens: a linear and

a quadratic increase up to $\lambda_{\max}$ at training epoch $T_{\max}$, as shown below with e the training epoch:

$$\lambda_{\text{sparse}}(e) = \begin{cases} \lambda_{max} & \text{flat regimen} \\ \min\{(e/T_{\max}) \cdot \lambda_{max}, \lambda_{max}\} & \text{linear regimen} \\ \min\{(e/T_{\max})^2 \cdot \lambda_{max}, \lambda_{max}\} & \text{quadratic regimen} \end{cases} .$$

4.2 Experiment

In this section I describe the experimental setup and specific instantiation of **Occam's Gates** for the three tasks.

4.2.1 Sentiment Analysis

The goal in this task is to produce the correct sentiment label from a sequence of words.

4.2.1.1 Dataset

I perform my experiment on the *Stanford Sentiment Treebank*, described earlier (**Section 3.2**). The dataset is composed of 11,855 syntactic parse trees with sentiment annotation at every node in the tree. To convert these trees to word sequences, I extract all possible word spans from the trees, forming 215,154 unique sub-phrases. The phrases are then split into training, cross validation, and testing using the provided *train-validation-test* split.

4.2.1.2 Implementation

Each word is projected using an embedding matrix into a 100 dimensional vector. Only words that appear at least twice in the training data are kept, with others replaced by an unknown word symbol, `<UNK>`.

The words vectors are read in sequence by a gated LSTM; I obtain a 5 class sentiment distribution after projecting the last hidden vector through an affine map and applying exponential normalization (Softmax). 3 different models are trained with different hidden sizes: 25, 50, and 150. Dropout with probability $p = 0.3$ is applied between the non recurrent connections of the LSTM [90, 116]. All models are trained using Adadelta [117] with $\rho = 0.95$, and training is stopped early when prediction accuracy stops increasing on the validation set.

The base error function to train this model is the Kullback-Leibler divergence between the correct sentiment label and the correct label. I apply Occam's Gates by adding the sum of the gate activations to the objective function λ_{sparse} (**equation 4.1**) (**Table 4.1**).

4.2.2 Paraphrase Detection

In Paraphrase Recognition the problem is it to predict whether two phrases carry the same meaning or are unrelated. This task can either be seen as regression or binary classification, and the goal is measured as the Pearson correlation with human annotations or recalling correct paraphrase pairs.

4.2.2.1 Dataset

Here, I focus on paraphrase detection on the SemEval 2014 shared task 1 dataset [62] which includes 9927 sentence pairs in a 4500/500/4927 *train-validation-test* split. Each sentence is annotated with a score $t \in [1, 5]$, with 5 indicating the pair is a paraphrase, and 1 that the pair is unrelated. I also train using paraphrase pairs from the wikianswers paraphrase corpus [24].

4.2.2.2 Implementation

For paraphrase prediction, I employ the same setup as for sentiment analysis (**Section 4.2.1**), with the final softmax layer removed. Each sentence in a pair is given to a different copy of the Gated LSTM.

The model's error function is the squared difference between the true similarity $t \in \mathbb{R}$ of the sentences and the dot product of the two LSTMs' final hidden states $\vec{h}_1, \vec{h}_2$, with $g_{1,i}$ the gates in sentence 1 and $g_{2,i}$ the gates in sentence 2:

$$J = \left(\frac{\vec{h}_1^T \vec{h}_2}{|\vec{h}_1||\vec{h}_2|} - t \right)^2 + \lambda_{\text{sparse}} \cdot \left(\sum_{i=1}^{n} g_{1,i} + \sum_{i=1}^{n} g_{2,i} \right).$$

4.2.3 Question Answering

4.2.3.1 Dataset

In this experiment we[1] focus on the 20 tasks in the Facebook bAbI dataset [108], which test different capabilities of a model, from coreference to logical induction, and use synthetic stories with limited vocabulary. We use the first 850 examples for training and remaining 150 for validation as suggested in [108]. Each story is made of several delimited sentences, and multiple questions along with their correct answer. Each question also contains a list of sentences from the story that contain information necessary to answer the question.

4.2.3.2 Implementation

For this task we construct a model that makes use of sparsity using the fact supervision information from bAbI, and by penalizing the activation of input gates controlling word vector inputs. We make use of the additional fact supervision information by using two separate Gated LSTMs for different levels of reader:

1. We use a separate **fact reader** for each sentence, and gate the word vectors. The final hidden state of this reader is kept as a *fact summary.*

2. A **story reader** receives as input a sequence of *fact summaries* from the **fact reader**. The gate from this reader controls the arrival of *fact summaries.*

The *fact reader*'s gates are unsupervised, and we apply Occam's Gates using λ_{sparse} as done for Sentiment Analysis and Paraphrase Prediction. The *story reader*, on the

[1] Joint work with Szymon Sidor.

other hand, has supervision on which facts should be used to answer a particular question. Using this information we penalize the log loss between the gate activations, g_is, and the supervision labels t_is, weighed by λ_{fact}:

$$L_{\text{supervision}}(g) = -\lambda_{\text{fact}} \frac{1}{n} \sum_{i=1}^{n} g_i \log(t_i) + (1 - g_i) \log(1 - t_i).$$

To train our model we project the final hidden state of the *story reader* into a distribution over all the possible answer words, and minimize the negative log likelihood of the correct answer under our model. Our final error function for this task is the sum of gate supervision error, $L_{\text{supervision}}$, the negative log likelihood for the correct answer, and Occam's Gates.

Through hyper parameter search on the validation set, we find that a model with 30 memory units in each LSTM, with 6 layers each, using quadratic gates (4.3), performs best. We train our model using Stochastic Gradient Descent using the AdaDelta update rule [117], with $\rho = 0.95$, and a minibatch size of 50. We perform early stopping when the validation score stops increasing.

4.3 Results

4.3.1 Effects on performance

Occam's Gates improve generalization on Sentiment Analysis (**Figure 4-4**), Paraphrase Prediction (**Figure 4-6**), and for 18 out of 20 bAbI question answering tasks (**Figure 4-8**), (**Table 4.2**). This effect is especially visible as model size increases (**Figure 4-4**), (**Figure 4-6**). We find that without a sparsity penalty increasing model size has smaller effect, however using sparsity we manage to achieve 5% improvement on sentiment analysis and 18% on paraphrase prediction recall. Additionally for *three arg. relations* bAbI problem it increases the accuracy by 14%[2].

Moreover, the sparsity annealing methods described in section (**Section 4.1.2**)

[2]We observe greater improvements on this task than the other two; coincidently, this task has longer sentences, thus word gating has greater impact.

Table 4.2: Comparison of test accuracy using different models on bAbI dataset for various tasks from [108]. Models are (left to right): LSTM baseline from [108], Stacked Gated LSTMs, and Memory Networks. Best results for LSTMs shown in bold.

Task	**LSTMs**	**Gated LSTMs**	**MN**
single supporting fact	50	**100**	100
two supporting facts	20	**32**	100
three supporting facts	**20**	**20**	100
two arg relations	61	**77**	100
three arg relations	**70**	66	98
yes-no questions	48	**51**	100
counting	49	**76**	85
lists sets	45	**78**	91
simple negation	64	**70**	100
indefinite knowledge	44	**47**	98
basic-coreference	72	**89**	100
conjunction	74	**99**	100
compound-coreference	**94**	93	100
time reasoning	**27**	**27**	99
basic deduction	21	**50**	100
basic induction	23	**47**	100
positional reasoning	51	**58**	65
size reasoning	52	**90**	95
path finding	**8**	**8**	36
agents motivations	91	**96**	100

show improvements over a static objective function (**Figure 4-5**) and (**Figure 4-7**). In particular, the linear regimen improves recall by 1% for Sentiment Analysis, and by 7% on paraphrase prediction.

Finally, we observed that the Gated-LSTMs using sparsity significantly improves performance over the LSTM baseline from [108]. As visible in table 4.2, this approach matches or outperforms a non sparse approach on 18 out of 20 problems. Moreover, in (**Figure 4-9**), in the absence of fact selection, word sparsity helps match the performance of true supervision. The proposed approach does not match the performance of Memory Networks (MemNN), however it allows inspection and visualization of the behavior of model when performing complex multi step tasks.

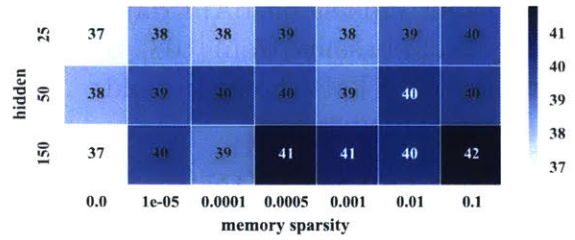

Figure 4-4: SST Root Accuracy with varying LSTM hidden size and sparsity penalty λ

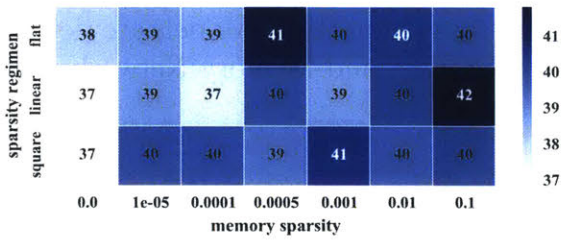

Figure 4-5: Effect of sparsity regimen and sparsity penalty λ on SST Root Accuracy.

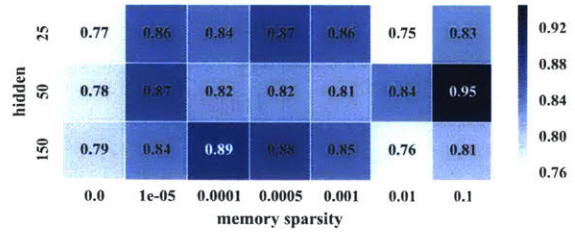

Figure 4-6: Paraphrase accuracy with varying LSTM hidden size and sparsity penalty λ

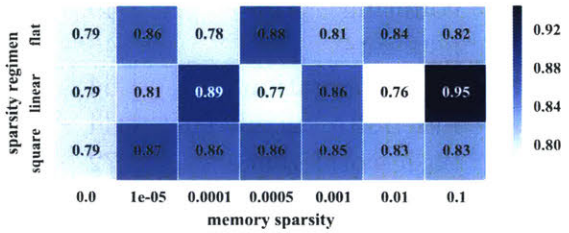

Figure 4-7: Effect of sparsity regimen and penalty λ on Paraphrase prediction.

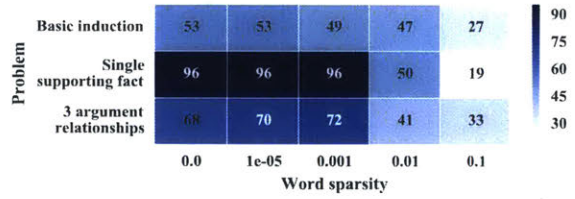

Figure 4-8: Accuracy for three bAbI tasks with varying λ_{word}

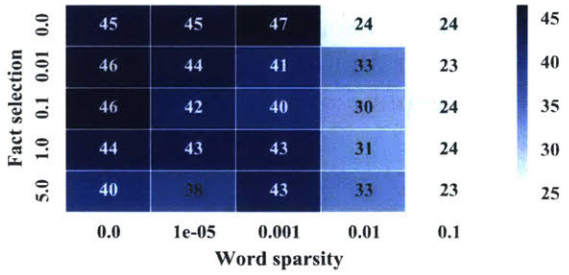

Figure 4-9: Effect of λ_{word} and λ_{fact} on Basic Induction task accuracy

4.3.1.1 Relevancy detection

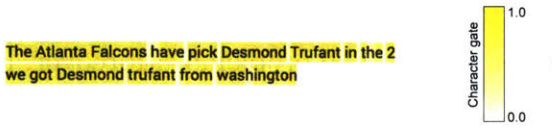

Figure 4-10: Gated LSTM viewing characters individually. Word boundaries are detected as shown by the absence of yellow highlighting in the inter-word spaces.

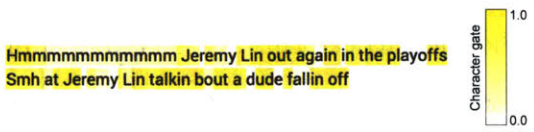

Figure 4-11: Gated LSTM viewing characters individually. Model focuses on upper case characters and ignores repeats.

A central goal of Occam's Gates is to apply a parsimony principle to any sequence to make the behavior of the network more transparent. A particularly salient example of the effect of this technique is visible when using a character based recurrent neural network. Under those conditions, the inputs at every time-step are word embeddings for every possible character (e.g. all ASCII characters). The resulting network when trained for paraphrase detection learns to detect word boundaries (**Figure 4-10**) and ignore repetitive or superfluous characters (**Figure 4-11**).

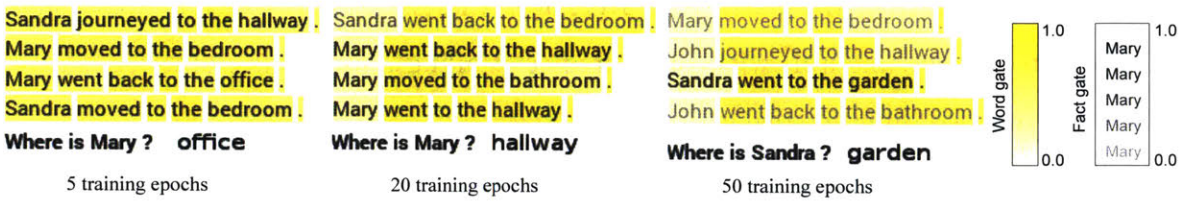

Figure 4-12: Gating procedure learnt by Gated LSTMs during training for bAbI task. We note here how the yellow word highlighting in the rightmost paragraph correspond to informative words, while in the leftmost paragraph no particular pattern is present. The text color indicates whether an entire sentence was kept using the fact gate. In the middle paragraph, sentences without the character mentioned in the question are already ignored.

A similar observation can be made regarding the search behavior discovered by the network when performing question answering tasks from bAbI. In this scenario, each sentence and word can be gated separately. Intuitively we would expect the network to discover how to ignore irrelevant sentences by shutting off the gate for the entire sentence, and for relevant sentences, shutting off irrelevant words. When viewing the evolution of the network during training in (**Figure 4-12**), a strategy emerges that is close to the one just described.

4.4 Analysis

In this chapter, I presented a complimentary objective function for training recurrent neural networks that renders their behavior more interpretable, while also improving the performance. In experiments with Occam's Gates, I demonstrated how the approach did not compromise performance for readability, and instead improved per-

formance over non-sparse baselines by a large margin.

In particular, enforcing sparsity reduced overfitting allowing models with more hidden units to be trained for Sentiment Analysis and Paraphrase Detection. In 18 of the 20 bAbI question answering tasks (**Table 4.2**), Gated-LSTMs trained with Occam's Gates improved over the non-sparse baseline.

Varying the sparsity penalty during training to allow early exploration proved beneficial by up to 7% on Paraphrase Detection recall (**Figure 4-7**), and 1.5% on Sentiment Analysis (**Figure 4-5**). Moreover, over- and under-penalization of sparsity are shown to be detrimental, with performance peaking at a single point ($\lambda_{\text{sparse}} = 0.0005$) during two different tasks.

Chapter 5

Discussion and Future Work

In this thesis, I provide evidence that in the Man-Machine interaction problem enforcing sparsity in the inputs of the machine sheds light on the decision process of the system and provides a rationale for the human. In this conclusion chapter, I will first summarize the findings, and second discuss areas of future work.

5.1 Conclusions

Through three separate experiments I find that mixed initiative systems can be more transparent and achieve higher performance by enforcing sparsity, either through the use of an Ontological User Profile that respects the hierarchical nature of the options under recommendation, or by adopting a new objective function to train Recurrent Neural Networks (RNNs) that induces an Occam's Razor behavior in their treatment of input.

5.1.1 Ontological User Profile

Semantic Relaxation Uhura is an extension to the present Uhura mixed-initiative planning system, which includes an Ontological User Profile that suggests alternative relaxation solutions from a user's preferences. Unlike competing approaches, this User Profile works under *cold* and *warm start* situations and respects the hierarchical

nature of the options under recommendation. The addition of an ontology improves the transparency of the system by allowing it to justify its ranking through the use of supporting examples in past user actions.

This richer representation also makes the final system more expressive: connections between user actions are found "semantically" by considering whether two choice nodes in the ontology share parents or siblings. Furthermore, by incorporating online reviews and metadata into the semantic distance, new choices can be dynamically incorporated into the recommendation by re-using the previously learnt topic model on new data.

I demonstrated the effectiveness and validity of the approach empirically by comparing the recommendations by humans with those made by the Semantic Relaxation Uhura or those made with other state of the art topic modelling methods. In this experiment, I find that using an Ontological User Profiles generated by reading online reviews improves agreement with humans by 6.4% (from 36.11% to 43.52%) over competing approaches.

5.1.2 Occam's Gates

Occam's Gates is an augmented objective function for training RNNs to be selective in their inputs that can be systematically applied to sequence learning tasks. Through two separate experiments, I demonstrate the effectiveness of the approach to provide justification for the machine's behavior, and improved performance.

5.1.2.0.1 Sentiment Analysis

Representing sentiment as an incremental, justifiable process, with each timestep's prediction updating the current best prediction rather than replacing it, gives a visual explanation for the regions of the input that contain emotional signal and increases prediction accuracy. The proposed model, which uses incremental gated predictions, outperforms a baseline by 7.03% on 5-class accuracy and 5.73% on binary accuracy, and improves over the current state of the art by 1.3% on 5-class accuracy, and 0.83%

on binary accuracy.

5.1.2.0.2 Sequence Learning

Occam's Gates (OG) can be systemically applied to other problems: I contrast non-gated approaches with their OG counterparts. In my experiments, I find that all models where OG is applied are now capable of pinpointing the temporal regions that impacted their prediction, and consistently outperform versions where sparsity was not enforced:

- **Sentiment Analysis**: A 5% improvement is observed over non-sparse models. The performance difference grows with the number of parameters in the model, suggesting that OG acts as an implicit regularizer by communicating that only a subset of the temporal inputs should impact the prediction.

- **Paraphrase Detection**: An 18% improvement is observed on Paraphrase Prediction recall. As with Sentiment Analysis, performance improves with model size, reinforcing the notion that OG also regularizes.

- **Question Answering and Fact Reading**: (bAbI dataset) LSTMs trained with OG outperform the non-gated LSTMs on 17 out of the 20 tasks. Moreover, on certain tasks, LSTM performance matches that of the more advanced Memory Networks, through the simple addition of a sparsity penalty.

5.2 Future Work

Greater transparency in Man-Machine interaction raises several questions about future division of labor. In the three experiments presented in this thesis, the ability to visualize, interact, and collaborate more deeply with the machine has significant effects on the usability of the mixed initiative systems and improves performance. In this section, I discuss several directions for future work with respect to the Ontological User Profile and Occam's Gates.

5.2.1 Ontological User Profile

Semantic Relaxation Uhura paves the way towards more dynamic objective functions within plan relaxation, where the user's preferences can not only allow new options to be integrated in the plan, but also modify the importance of difference constraints. In this research, we demonstrate the feasibility of a dynamic objective function expanding the set of relaxation solutions from CDRU.

In the future, there would be significant value in understanding to what extent other parts of the objective function could become dynamic based on the environment or user preferences, by integrating multiple user preferences jointly as was done in Conditional-Preference networks [12, 17].

5.2.2 Occam's Gates

In this thesis I provide evidence that Occam's Gates (OG) can be systematically applied to several sequence learning tasks to construct systems that are more transparent to the human actor, and more performant by reducing the impact of distractors.

In fact, the experiments appear to confirm that this augmented objective function communicates problem structure: I find that by modifying a baseline model, a bidirectional LSTM, so that it incrementally constructs its prediction using gates, I outperform more sophisticated approaches. This result suggests that OG could provide similar benefits if combined with different models or if directed at other sequence learning tasks. With this observation in mind, there exists several directions where OG could be further investigated:

- Computer Vision models, which make use of Recurrent Neural Networks to control a sequence of glimpses [29, 95, 66, 46, 104, 119, 85, 77] through images, could similarly be made more robust to distractors by applying OG.

- Mixed initiative architectures contain several black-box components responsible for recognizing state or activities from sequential data where sparse gates could help detect failures or communicate information flow to the human.

Bibliography

[1] Final Report: Flight A.F. 447 Rio de Janeiro - Paris. Technical Report F-GZCP, Burean d'Enquêtes et d'Analyses pour la sécurité de l'aviation civile, *Bâtiment 153, 10 rue de Paris, Zone Sud, Aéroport du Bourget, 93352 Le Bourget, France*, July 2012.

[2] James F Allen and Lenhart K Schubert. The trains project. Technical report, DTIC Document, 1991.

[3] Nikolay Archak, Anindya Ghose, and Panagiotis G Ipeirotis. Show me the money!: deriving the pricing power of product features by mining consumer reviews. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 56–65. ACM, 2007.

[4] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[5] Regina Barzilay. *Information fusion for multidocument summarization: paraphrasing and generation.* PhD thesis, Columbia University, 2003.

[6] Justin Bayer, Christian Osendorfer, Daniela Korhammer, Nutan Chen, Sebastian Urban, and Patrick van der Smagt. On fast dropout and its applicability to recurrent networks. *arXiv preprint arXiv:1311.0701*, 2013.

[7] James Bennett and Stan Lanning. The netflix prize. In *Proceedings of KDD cup and workshop*, volume 2007, page 35, 2007.

[8] Mark Bloom. Two astros walk on the moon in man's 1st step in universe. 21 July 1969. `http://www.nydailynews.com/news/world/apollo-11-lands-man-moon-1969-article-1.2294428`.

[9] Johan Bollen, Huina Mao, and Xiao-Jun Zeng. Twitter mood predicts the stock market. *CoRR*, abs/1010.3003, 2010.

[10] Antoine Bordes, Sumit Chopra, and Jason Weston. Question answering with subgraph embeddings. *arXiv preprint arXiv:1406.3676*, 2014.

[11] Antoine Bordes, Nicolas Usunier, Sumit Chopra, and Jason Weston. Large-scale simple question answering with memory networks. *arXiv preprint arXiv:1506.02075*, 2015.

[12] Craig Boutilier, Ronen I. Brafman, Carmel Domshlak, Holger H. Hoos, and David Poole. Cp-nets: A tool for representing and reasoning with conditional ceteris paribus preference statements. *J. Artif. Intell. Res.(JAIR)*, 21:135–191, 2004.

[13] Mitch Bryson, Alistair Reid, Fabio Ramos, and Salah Sukkarieh. Airborne vision-based mapping and classification of large farmland environments. *Journal of Field Robotics*, 27(5):632–655, 2010.

[14] Sean Burke, Enrique Fernandez, Luis Figueredo, Andreas Hofmann, Christopher Hofmann, Erez Karpas, Steve Levine, Pedro Santana, Peng Yu, and Brian Williams. Intent recognition and temporal relaxation in human robot assembly. In *Proceedings of the Twenty-fourth International Conference on Automated Planning and Scheduling (ICAPS-14)*, 2014.

[15] Jaime R Carbonell. Ai in cai: An artificial-intelligence approach to computer-assisted instruction. *Man-Machine Systems, IEEE Transactions on*, 11(4):190–202, 1970.

[16] Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. Natural language processing (almost) from scratch. *The Journal of Machine Learning Research*, 12:2493–2537, 2011.

[17] Cristina Cornelio, Judy Goldsmith, Nicholas Mattei, Francesca Rossi, and K Brent Venable. Dynamic and probabilistic cp-nets. *CP Doctoral Program 2013*, page 31, 2012.

[18] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[19] Roddy Cowie, Ellen Douglas-Cowie, Nicolas Tsapatsoulis, George Votsis, Stefanos Kollias, Winfried Fellenz, and John G Taylor. Emotion recognition in human-computer interaction. *Signal Processing Magazine, IEEE*, 18(1):32–80, 2001.

[20] Guilelmus de Ockham. *Summa totius Logicae.*

[21] Rina Dechter, Itay Meiri, and Judea Pearl. Temporal constraint networks. *Artificial intelligence*, 49(1):61–95, 1991.

[22] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *The Journal of Machine Learning Research*, 12:2121–2159, 2011.

[23] Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A. Smith. Transition-based dependency parsing with stack long short-term memory. *CoRR*, abs/1505.08075, 2015.

[24] Anthony Fader, Luke S Zettlemoyer, and Oren Etzioni. Paraphrase-driven learning for open question answering. In *ACL (1)*, pages 1608–1618, 2013.

[25] Vittorio Gallese, Luciano Fadiga, Leonardo Fogassi, and Giacomo Rizzolatti. Action recognition in the premotor cortex. *Brain*, 119(2):593–609, 1996.

[26] F. A. Gers. *Long Short-Term Memory in Recurrent Neural Networks.* PhD thesis, Swiss Federal Institute of Technology, Department of Computer Science, Lausanne, EPFL, Switzerland, 2001.

[27] Alex Graves. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*, 2013.

[28] Klaus Greff, Rupesh Kumar Srivastava, Jan Koutník, Bas R. Steunebrink, and Jürgen Schmidhuber. LSTM: A search space odyssey. *CoRR*, abs/1503.04069, 2015.

[29] Karol Gregor, Ivo Danihelka, Alex Graves, and Daan Wierstra. Draw: A recurrent neural network for image generation. *arXiv preprint arXiv:1502.04623*, 2015.

[30] Joshua Hailpern and Bernardo Huberman. Presenting documents to a user based on topics and collective opinions expressed in the documents, December 17 2012. US Patent App. 13/717,151.

[31] Joshua Hailpern and Bernardo A Huberman. Echo: the editor's wisdom with the elegance of a magazine. In *Proceedings of the 5th ACM SIGCHI symposium on Engineering interactive computing systems*, pages 313–322. ACM, 2013.

[32] Javier Hernandez, Mohammed Ehsan Hoque, Will Drevo, and Rosalind W Picard. Mood meter: counting smiles in the wild. In *Proceedings of the 2012 ACM Conference on Ubiquitous Computing*, pages 301–310. ACM, 2012.

[33] Geoffrey E. Hinton and Ruslan R Salakhutdinov. Replicated softmax: an undirected topic model. In Y. Bengio, D. Schuurmans, J.D. Lafferty, C.K.I. Williams, and A. Culotta, editors, *Advances in Neural Information Processing Systems 22*, pages 1607–1614. Curran Associates, Inc., 2009.

[34] Jerry R Hobbs, Douglas Appelt Sr, John S Bear, David Israel Sr, and W Mabry Tyson. Fastus: A system for extracting information from natural-language text. Technical report, DTIC Document, 1992.

[35] Sepp Hochreiter, Yoshua Bengio, Paolo Frasconi, and Jürgen Schmidhuber. Gradient flow in recurrent nets: the difficulty of learning long-term dependencies, 2001.

[36] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[37] Thomas Hofmann. Probabilistic latent semantic analysis. In *Proceedings of the Fifteenth conference on Uncertainty in artificial intelligence*, pages 289–296. Morgan Kaufmann Publishers Inc., 1999.

[38] Eric Horvitz and Matthew Barry. Display of information for time-critical decision making. In *Proceedings of the Eleventh conference on Uncertainty in artificial intelligence*, pages 296–305. Morgan Kaufmann Publishers Inc., 1995.

[39] Eric Horvitz, Jack Breese, David Heckerman, David Hovel, and Koos Rommelse. The lumiere project: Bayesian user modeling for inferring the goals and needs of software users. In *Proceedings of the Fourteenth conference on Uncertainty in artificial intelligence*, pages 256–265. Morgan Kaufmann Publishers Inc., 1998.

[40] Ozan Irsoy and Ethem Alpaydın. Autoencoder trees. *arXiv preprint arXiv:1409.7461*, 2014.

[41] Ozan Irsoy and Claire Cardie. Bidirectional recursive neural networks for token-level labeling with structure. *arXiv preprint arXiv:1312.0493*, 2013.

[42] Ozan Irsoy and Claire Cardie. Deep recursive neural networks for compositionality in language. In *Advances in Neural Information Processing Systems*, pages 2096–2104, 2014.

[43] Ozan İrsoy and Claire Cardie. Modeling compositionality with multiplicative recurrent neural networks. *arXiv preprint arXiv:1412.6577*, 2014.

[44] Sébastien Jean, Kyunghyun Cho, Roland Memisevic, and Yoshua Bengio. On using very large target vocabulary for neural machine translation. *arXiv preprint arXiv:1412.2007*, 2014.

[45] Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. A convolutional neural network for modelling sentences. *arXiv preprint arXiv:1404.2188*, 2014.

[46] Andrej Karpathy and Li Fei-Fei. Deep visual-semantic alignments for generating image descriptions. *arXiv preprint arXiv:1412.2306*, 2014.

[47] Boris Katz. Annotating the world wide web using natural language. In *RIAO*, pages 136–159, 1997.

[48] Boris Katz, Gary C Borchardt, and Sue Felshin. Natural language annotations for question answering. In *FLAIRS Conference*, pages 303–306, 2006.

[49] Boris Katz, Sue Felshin, Deniz Yuret, Ali Ibrahim, Jimmy Lin, Gregory Marton, Alton Jerome McFarland, and Baris Temelkuran. Omnibase: Uniform access to heterogeneous data for question answering. In *Natural Language Processing and Information Systems*, pages 230–234. Springer, 2002.

[50] Charles Kemp and Joshua B. Tenenbaum. The discovery of structural form. *Proceedings of the National Academy of Sciences*, 105(31):10687–10692, 2008.

[51] Charles Kemp and Joshua B. Tenenbaum. Structured statistical models of inductive reasoning. *Psychological review*, 116(1):20, 2009.

[52] Ryan Kiros, Yukun Zhu, Ruslan Salakhutdinov, Richard S Zemel, Antonio Torralba, Raquel Urtasun, and Sanja Fidler. Skip-thought vectors. *arXiv preprint arXiv:1506.06726*, 2015.

[53] Douglas J. Klein and M Randić. Resistance distance. *Journal of Mathematical Chemistry*, 12(1):81–95, 1993.

[54] Maciej Kula. Metadata embeddings for user and item cold-start recommendations. *CoRR*, abs/1507.08439, 2015.

[55] Ankit Kumar, Ozan Irsoy, Jonathan Su, James Bradbury, Robert English, Brian Pierce, Peter Ondruska, Ishaan Gulrajani, and Richard Socher. Ask me anything: Dynamic memory networks for natural language processing. *arXiv preprint arXiv:1506.07285*, 2015.

[56] Hugo Larochelle and Stanislas Lauly. A neural autoregressive topic model. In F. Pereira, C.J.C. Burges, L. Bottou, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 2708–2716. Curran Associates, Inc., 2012.

[57] Quoc V Le and Tomas Mikolov. Distributed representations of sentences and documents. *arXiv preprint arXiv:1405.4053*, 2014.

[58] Louis Legendre and L Legrendre. *Numerical ecology. Developments in environmental modelling.* Elsevier Science & Technology, 1998.

[59] Percy Liang and Christopher Potts. Bringing machine learning and compositional semantics together. *Annu. Rev. Linguist.*, 1(1):355–376, 2015.

[60] Bing Liu, Minqing Hu, and Junsheng Cheng. Opinion observer: analyzing and comparing opinions on the web. In *Proceedings of the 14th international conference on World Wide Web*, pages 342–351. ACM, 2005.

[61] Minh-Thang Luong, Ilya Sutskever, Quoc V Le, Oriol Vinyals, and Wojciech Zaremba. Addressing the rare word problem in neural machine translation. In *Proceedings of ACL*, 2015.

[62] Marco Marelli, Luisa Bentivogli, Marco Baroni, Raffaella Bernardi, Stefano Menini, and Roberto Zamparelli. Semeval-2014 task 1: Evaluation of compositional distributional semantic models on full sentences through semantic relatedness and textual entailment. *SemEval-2014*, 2014.

[63] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[64] Tomas Mikolov, Quoc V Le, and Ilya Sutskever. Exploiting similarities among languages for machine translation. *arXiv preprint arXiv:1309.4168*, 2013.

[65] Jeff Mitchell and Mirella Lapata. Composition in distributional models of semantics. *Cognitive science*, 34(8):1388–1429, 2010.

[66] Volodymyr Mnih, Nicolas Heess, Alex Graves, et al. Recurrent models of visual attention. In *Advances in Neural Information Processing Systems*, pages 2204–2212, 2014.

[67] Derek Monner and James A Reggia. A generalized lstm-like training algorithm for second-order recurrent neural networks. *Neural Networks*, 25:70–83, 2012.

[68] Robert E Morin, Bert Forrin, and Wayne Archer. Information processing behavior: The role of irrelevant stimulus information. *Journal of Experimental Psychology*, 61(1):89, 1961.

[69] David G Novick and Stephen Sutton. What is mixed-initiative interaction. In *Procedings of the AAAI Spring Symposium on Computational Models for Mixed Initiative Interaction*, pages 114–116, 1997.

[70] Larry Page, S Brin, R Motwani, and T Winograd. Pagerank: Bringing order to the web. Technical report, Stanford Digital Libraries Working Paper, 1997.

[71] Bo Pang and Lillian Lee. Opinion mining and sentiment analysis. *Foundations and trends in information retrieval*, 2(1-2):1–135, 2008.

[72] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. *arXiv preprint arXiv:1211.5063*, 2012.

[73] James G Paterson, Eric Timmons, and Brian C. Williams. A scheduler for actions with iterated durations. In *Twenty-Eighth AAAI Conference on Artificial Intelligence*, 2014.

[74] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. *Proceedings of the Empiricial Methods in Natural Language Processing (EMNLP 2014)*, 12:1532–1543, 2014.

[75] Vu Pham, Théodore Bluche, Christopher Kermorvant, and Jérôme Louradour. Dropout improves recurrent neural networks for handwriting recognition. *arXiv preprint arXiv:1312.4569*, 2013.

[76] Jonathan Raiman and Szymon Sidor. Occam's gates. *arXiv preprint arXiv:1506.08251*, 2015.

[77] Marc'Aurelio Ranzato. On learning where to look. *arXiv preprint arXiv:1405.5488*, 2014.

[78] Matthew Richardson, Christopher JC Burges, and Erin Renshaw. Mctest: A challenge dataset for the open-domain machine comprehension of text. In *EMNLP*, volume 1, page 2, 2013.

[79] Giacomo Rizzolatti and Maddalena Fabbri-Destro. Mirror neurons: from discovery to autism. *Experimental Brain Research*, 200(3-4):223–237, 2010.

[80] Tim Rocktäschel, Edward Grefenstette, Karl Moritz Hermann, Tomáš Kočiskỳ, and Phil Blunsom. Reasoning about entailment with neural attention. *arXiv preprint arXiv:1509.06664*, 2015.

[81] Gerard Salton and Christopher Buckley. Term-weighting approaches in automatic text retrieval. *Information processing & management*, 24(5):513–523, 1988.

[82] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. Analysis of recommendation algorithms for e-commerce. In *Proceedings of the 2nd ACM conference on Electronic commerce*, pages 158–167. ACM, 2000.

[83] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international conference on World Wide Web*, pages 285–295. ACM, 2001.

[84] Iulian V Serban, Alessandro Sordoni, Yoshua Bengio, Aaron Courville, and Joelle Pineau. Hierarchical neural network generative models for movie dialogues. *arXiv preprint arXiv:1507.04808*, 2015.

[85] Pierre Sermanet, Andrea Frome, and Esteban Real. Attention for fine-grained categorization. *arXiv preprint arXiv:1412.7054*, 2014.

[86] Richard Socher, Jeffrey Pennington, Eric H. Huang, Andrew Y Ng, and Christopher D Manning. Semi-supervised recursive autoencoders for predicting sentiment distributions. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 151–161. Association for Computational Linguistics, 2011.

[87] Richard Socher, Alex Perelygin, Jean Y Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the conference on empirical methods in natural language processing (EMNLP)*, volume 1631, page 1642, 2013.

[88] Robert R Sokal. A statistical method for evaluating systematic relationships. *Univ Kans Sci Bull*, 38:1409–1438, 1958.

[89] Alessandro Sordoni, Michel Galley, Michael Auli, Chris Brockett, Yangfeng Ji, Margaret Mitchell, Jian-Yun Nie, Jianfeng Gao, and Bill Dolan. A neural network approach to context-sensitive generation of conversational responses. *arXiv preprint arXiv:1506.06714*, 2015.

[90] Nitish Srivastava. *Improving neural networks with dropout.* PhD thesis, University of Toronto, 2013.

[91] Kazunari Sugiyama, Kenji Hatano, and Masatoshi Yoshikawa. Adaptive web search based on user profile constructed without any effort from users. In *Proceedings of the 13th international conference on World Wide Web*, pages 675–684. ACM, 2004.

[92] Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston, and Rob Fergus. End-to-end memory networks. In *Advances in Neural Information Processing Systems*, 2015.

[93] Ilya Sutskever, Oriol Vinyals, and Quoc VV Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112, 2014.

[94] Kai Sheng Tai, Richard Socher, and Christopher D Manning. Improved semantic representations from tree-structured long short-term memory networks. *arXiv preprint arXiv:1503.00075*, 2015.

[95] Yichuan Tang, Nitish Srivastava, and Ruslan R Salakhutdinov. Learning generative models with visual attention. In Z. Ghahramani, M. Welling, C. Cortes, N.D. Lawrence, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 1808–1816. Curran Associates, Inc., 2014.

[96] Joshua B. Tenenbaum, Thomas L. Griffiths, and Charles Kemp. Theory-based bayesian models of inductive learning and reasoning. *Trends in cognitive sciences*, 10(7):309–318, 2006.

[97] Joshua B. Tenenbaum, Charles Kemp, Thomas L. Griffiths, and Noah D. Goodman. How to grow a mind: Statistics, structure, and abstraction. *science*, 331(6022):1279–1285, 2011.

[98] Tomer D. Ullman, Noah D. Goodman, and Joshua B. Tenenbaum. Theory learning as stochastic search in the language of thought. *Cognitive Development*, 27(4):455–480, 2012.

[99] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(2579-2605):85, 2008.

[100] Thierry Vidal. Handling contingency in temporal constraint networks: from consistency to controllabilities. *Journal of Experimental & Theoretical Artificial Intelligence*, 11(1):23–45, 1999.

[101] Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. Pointer networks. *arXiv preprint arXiv:1506.03134*, 2015.

[102] Oriol Vinyals, Lukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey Hinton. Grammar as a foreign language. *arXiv preprint arXiv:1412.7449*, 2014.

[103] Oriol Vinyals and Quoc Le. A neural conversational model. *arXiv preprint arXiv:1506.05869*, 2015.

[104] Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. Show and tell: A neural image caption generator. *arXiv preprint arXiv:1411.4555*, 2014.

[105] Ulrike Von Luxburg. A tutorial on spectral clustering. *Statistics and computing*, 17(4):395–416, 2007.

[106] Ellen M Voorhees and Dawn M Tice. Building a question answering test collection. In *Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 200–207. ACM, 2000.

[107] Hao Wang, Dogan Can, Abe Kazemzadeh, François Bar, and Shrikanth Narayanan. A system for real-time twitter sentiment analysis of 2012 us presidential election cycle. In *Proceedings of the ACL 2012 System Demonstrations*, pages 115–120. Association for Computational Linguistics, 2012.

[108] Jason Weston, Antoine Bordes, Sumit Chopra, and Tomas Mikolov. Towards AI-complete question answering: A set of prerequisite toy tasks. *CoRR*, abs/1502.05698, 2015.

[109] Jason Weston, Antoine Bordes, Sumit Chopra, and Tomas Mikolov. Towards ai-complete question answering: A set of prerequisite toy tasks. *arXiv preprint arXiv:1502.05698*, 2015.

[110] Jason Weston, Sumit Chopra, and Antoine Bordes. Memory networks. *CoRR*, abs/1410.3916, 2014.

[111] Ryen W White, Nicholas P Tatonetti, Nigam H. Shah, Russ B Altman, and Eric Horvitz. Web-scale pharmacovigilance: listening to signals from the crowd. *Journal of the American Medical Informatics Association*, 20(3):404–408, 2013.

[112] Brian C. Williams and Robert Ragno. Conflict-directed a* and its role in model-based embedded systems. *Special Issue on Theory and Applications of Satisfiability Testing, Journal of Discrete Applied Math*, 155(12):1562–1595, jun 2003.

[113] Brian C Williams and Robert J Ragno. Conflict-directed a* and its role in model-based embedded systems. In *Journal of Discrete Applied Mathematics*. Citeseer, 2003.

[114] Peng Yu, Cheng Fang, and Brian Williams. Resolving uncontrollable conditional temporal problems using continuous relaxations. In *Proceedings of the Twenty-fourth International Conference on Automated Planning and Scheduling (ICAPS-14)*, 2014.

[115] Peng Yu and Brian Williams. Continuously relaxing over-constrained conditional temporal problems through generalized conflict learning and resolution. In *Proceedings of the 23th International Joint Conference on Artificial Intelligence (IJCAI-13)*, pages 2429–2436, Beijing, China, 2013.

[116] Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. Recurrent neural network regularization. *arXiv preprint arXiv:1409.2329*, 2014.

[117] Matthew D Zeiler. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012.

[118] Dell Zhang, Jun Wang, Deng Cai, and Jinsong Lu. Self-taught hashing for fast similarity search. In *Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval*, pages 18–25. ACM, 2010.

[119] Yin Zheng, Richard S Zemel, Yu-Jin Zhang, and Hugo Larochelle. A neural autoregressive approach to attention-based recognition. *International Journal of Computer Vision*, pages 1–13, 2014.

[120] Xiaojin Zhu, John D Lafferty, and Zoubin Ghahramani. Semi-supervised learning: From gaussian fields to gaussian processes. Technical Report 8-2003, *Carnegie Mellon University*, 2003.