Practical Artificial Intelligence Programming With Java

Third Edition

Mark Watson

November 11, 2008

Contents

Preface **xi**

1 Introduction **1**
1.1 Other JVM Languages 1
1.2 Why is a PDF Version of this Book Available Free on the Web? . . . 1
1.3 Book Software 2
1.4 Use of Java Generics and Native Types 2
1.5 Notes on Java Coding Styles Used in this Book 3
1.6 Book Summary 4

2 Search **5**
2.1 Representation of Search State Space and Search Operators 5
2.2 Finding Paths in Mazes 6
2.3 Finding Paths in Graphs 13
2.4 Adding Heuristics to Breadth First Search 22
2.5 Search and Game Playing 22
2.5.1 Alpha-Beta Search 22
2.5.2 A Java Framework for Search and Game Playing 24
2.5.3 Tic-Tac-Toe Using the Alpha-Beta Search Algorithm 29
2.5.4 Chess Using the Alpha-Beta Search Algorithm 34

3 Reasoning **45**
3.1 Logic 46
3.1.1 History of Logic 47
3.1.2 Examples of Different Logic Types 47
3.2 PowerLoom Overview 48
3.3 Running PowerLoom Interactively 49
3.4 Using the PowerLoom APIs in Java Programs 52
3.5 Suggestions for Further Study 54

4 Semantic Web **57**
4.1 Relational Database Model Has Problems Dealing with Rapidly Changing Data Requirements 58
4.2 RDF: The Universal Data Format 59
4.3 Extending RDF with RDF Schema 62
4.4 The SPARQL Query Language 63
4.5 Using Sesame 67

4.6 OWL: The Web Ontology Language 69
4.7 Knowledge Representation and REST 71
4.8 Material for Further Study 72

5 Expert Systems 73
5.1 Production Systems 75
5.2 The Drools Rules Language 75
5.3 Using Drools in Java Applications 77
5.4 Example Drools Expert System: Blocks World 81
5.4.1 POJO Object Models for Blocks World Example 82
5.4.2 Drools Rules for Blocks World Example 85
5.4.3 Java Code for Blocks World Example 88
5.5 Example Drools Expert System: Help Desk System 90
5.5.1 Object Models for an Example Help Desk 91
5.5.2 Drools Rules for an Example Help Desk 93
5.5.3 Java Code for an Example Help Desk 95
5.6 Notes on the Craft of Building Expert Systems 97

6 Genetic Algorithms 99
6.1 Theory 99
6.2 Java Library for Genetic Algorithms 101
6.3 Finding the Maximum Value of a Function 105

7 Neural Networks 109
7.1 Hopfield Neural Networks 110
7.2 Java Classes for Hopfield Neural Networks 111
7.3 Testing the Hopfield Neural Network Class 114
7.4 Back Propagation Neural Networks 116
7.5 A Java Class Library for Back Propagation 119
7.6 Adding Momentum to Speed Up Back-Prop Training 127

8 Machine Learning with Weka 129
8.1 Using Weka's Interactive GUI Application 130
8.2 Interactive Command Line Use of Weka 132
8.3 Embedding Weka in a Java Application 134
8.4 Suggestions for Further Study 136

9 Statistical Natural Language Processing 137
9.1 Tokenizing, Stemming, and Part of Speech Tagging Text 137
9.2 Named Entity Extraction From Text 141
9.3 Using the WordNet Linguistic Database 144
9.3.1 Tutorial on WordNet 144
9.3.2 Example Use of the JAWS WordNet Library 145
9.3.3 Suggested Project: Using a Part of Speech Tagger to Use the Correct WordNet Synonyms 149

9.3.4 Suggested Project: Using WordNet Synonyms to Improve Document Clustering . . . 150
9.4 Automatically Assigning Tags to Text . . . 150
9.5 Text Clustering . . . 152
9.6 Spelling Correction . . . 156
9.6.1 GNU ASpell Library and Jazzy . . . 157
9.6.2 Peter Norvig's Spelling Algorithm . . . 158
9.6.3 Extending the Norvig Algorithm by Using Word Pair Statistics 162
9.7 Hidden Markov Models . . . 166
9.7.1 Training Hidden Markov Models . . . 168
9.7.2 Using the Trained Markov Model to Tag Text . . . 173

10 Information Gathering 177
10.1 Open Calais . . . 177
10.2 Information Discovery in Relational Databases . . . 181
10.2.1 Creating a Test Derby Database Using the CIA World Fact-Book and Data on US States . . . 182
10.2.2 Using the JDBC Meta Data APIs . . . 183
10.2.3 Using the Meta Data APIs to Discern Entity Relationships . 187
10.3 Down to the Bare Metal: In-Memory Index and Search . . . 187
10.4 Indexing and Search Using Embedded Lucene . . . 193
10.5 Indexing and Search with Nutch Clients . . . 197
10.5.1 Nutch Server Fast Start Setup . . . 198
10.5.2 Using the Nutch OpenSearch Web APIs . . . 201

11 Conclusions 207

List of Figures

2.1 A directed graph representation is shown on the left and a two-dimensional grid (or maze) representation is shown on the right. In both representations, the letter R is used to represent the current position (or reference point) and the arrowheads indicate legal moves generated by a search operator. In the maze representation, the two grid cells marked with an X indicate that a search operator cannot generate this grid location. 7
2.2 UML class diagram for the maze search Java classes 8
2.3 Using depth first search to find a path in a maze finds a non-optimal solution 10
2.4 Using breadth first search in a maze to find an optimal solution . . . 14
2.5 UML class diagram for the graph search classes 15
2.6 Using depth first search in a sample graph 21
2.7 Using breadth first search in a sample graph 21
2.8 Alpha-beta algorithm applied to part of a game of tic-tac-toe 23
2.9 UML class diagrams for game search engine and tic-tac-toe 30
2.10 UML class diagrams for game search engine and chess 35
2.11 The example chess program does not contain an opening book so it plays to maximize the mobility of its pieces and maximize material advantage using a two-move lookahead. The first version of the chess program contains a few heuristics like wanting to control the center four squares. 36
2.12 Continuing the first sample game: the computer is looking ahead two moves and no opening book is used. 37
2.13 Second game with a 2 1/2 move lookahead. 41
2.14 Continuing the second game with a two and a half move lookahead. We will add more heuristics to the static evaluation method to reduce the value of moving the queen early in the game. 42

3.1 Overview of how we will use PowerLoom for development and deployment 46

4.1 Layers of data models used in implementing Semantic Web applications 58
4.2 Java utility classes and interface for using Sesame 68

5.1 Using Drools for developing rule-based systems and then deploying them. . . . 74
5.2 Initial state of a blocks world problem with three blocks stacked on top of each other. The goal is to move the blocks so that block C is on top of block A. . . . 82
5.3 Block C has been removed from block B and placed on the table. . . . 82
5.4 Block B has been removed from block A and placed on the table. . . . 84
5.5 The goal is solved by placing block C on top of block A. . . . 85

6.1 The test function evaluated over the interval [0.0, 10.0]. The maximum value of 0.56 occurs at x=3.8 . . . 100
6.2 Crossover operation . . . 101

7.1 Physical structure of a neuron . . . 110
7.2 Two views of the same two-layer neural network; the view on the right shows the connection weights between the input and output layers as a two-dimensional array. . . . 117
7.3 Sigmoid and derivative of the Sigmoid (SigmoidP) functions. This plot was produced by the file src-neural-networks/Graph.java. . . . 118
7.4 Capabilities of zero, one, and two hidden neuron layer neural networks. The grayed areas depict one of two possible output values based on two input neuron activation values. Note that this is a two-dimensional case for visualization purposes; if a network had ten input neurons instead of two, then these plots would have to be ten-dimensional instead of two-dimensional. . . . 119
7.5 Example backpropagation neural network with one hidden layer. . . . 120
7.6 Example backpropagation neural network with two hidden layers. . . . 120

8.1 Running the Weka Data Explorer . . . 131
8.2 Running the Weka Data Explorer . . . 131

List of Tables

2.1 Runtimes by Method for Chess Program 44

6.1 Random chromosomes and the floating point numbers that they encode 106

9.1 Most commonly used part of speech tags 139
9.2 Sample part of speech tags 167
9.3 Transition counts from the first tag (shown in row) to the second tag (shown in column). We see that the transition from NNP to VB is common. 169
9.4 Normalize data in Table 9.3 to get probability of one tag (seen in row) transitioning to another tag (seen in column) 171
9.5 Probabilities of words having specific tags. Only a few tags are shown in this table. 172