Deep Learning for Natural Language Processing

Deep learning is becoming increasingly important in a technology-dominated world. However, the building of computational models that accurately represent linguistic structures is complex, as it involves an in-depth knowledge of neural networks and the understanding of advanced mathematical concepts such as calculus and statistics. This book makes these complexities accessible to those from a humanities and social sciences background by providing a clear introduction to deep learning for natural language processing. It covers both theoretical and practical aspects and assumes minimal knowledge of machine learning, explaining the theory behind natural language in an easy-to-read way. It includes pseudo code for the simpler algorithms discussed and actual Python code for the more complicated architectures, using modern deep learning libraries such as PyTorch and Hugging Face. Providing the necessary theoretical foundation and practical tools, this book will enable readers to immediately begin building real-world, practical natural language processing systems.

M I H A I S U R D E A N U is an associate professor in the computer science department at the University of Arizona. He works in both academia and industry on natural language processing systems that process and extract meaning from natural language.

M A R C O A . V A L E N Z U E L A - E S C Á R C E G A is a research scientist in the computer science department at the University of Arizona. He has worked on natural language processing projects in both industry and academia.
Deep Learning for Natural Language Processing

A Gentle Introduction

Mihai Surdeanu
University of Arizona

Marco Antonio Valenzuela-Escárcega
University of Arizona
Contents

List of Figures \hspace{1cm} \textit{page x}

List of Tables \hspace{1cm} xv

Preface \hspace{1cm} xvii

1 Introduction 1
\hspace{1cm} 1.1 What This Book Covers 3
\hspace{1cm} 1.2 What This Book Does Not Cover 5
\hspace{1cm} 1.3 Deep Learning Is Not Perfect 5
\hspace{1cm} 1.4 Mathematical Notations 6

2 The Perceptron 8
\hspace{1cm} 2.1 Machine Learning Is Easy 8
\hspace{1cm} 2.2 Use Case: Text Classification 11
\hspace{1cm} 2.3 Evaluation Measures for Text Classification 12
\hspace{1cm} 2.4 The Perceptron 14
\hspace{1cm} 2.5 Voting Perceptron 22
\hspace{1cm} 2.6 Average Perceptron 24
\hspace{1cm} 2.7 Drawbacks of the Perceptron 26
\hspace{1cm} 2.8 Historical Background 28
\hspace{1cm} 2.9 References and Further Readings 29
\hspace{1cm} 2.10 Summary 29

3 Logistic Regression 30
\hspace{1cm} 3.1 The Logistic Regression Decision Function and Learning Algorithm 30
\hspace{1cm} 3.2 The Logistic Regression Cost Function 32
\hspace{1cm} 3.3 Gradient Descent 34
\hspace{1cm} 3.4 Deriving the Logistic Regression Update Rule 38
\hspace{1cm} 3.5 From Binary to Multiclass Classification 40
\hspace{1cm} 3.6 Evaluation Measures for Multiclass Text Classification 43
\hspace{1cm} v
<table>
<thead>
<tr>
<th></th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.7</td>
<td>Drawbacks of Logistic Regression 46</td>
</tr>
<tr>
<td>3.8</td>
<td>Historical Background 46</td>
</tr>
<tr>
<td>3.9</td>
<td>References and Further Readings 47</td>
</tr>
<tr>
<td>3.10</td>
<td>Summary 48</td>
</tr>
<tr>
<td>4</td>
<td>Implementing Text Classification Using Perceptron and Logistic Regression 49</td>
</tr>
<tr>
<td>4.1</td>
<td>Binary Classification 49</td>
</tr>
<tr>
<td>4.2</td>
<td>Multiclass Classification 62</td>
</tr>
<tr>
<td>4.3</td>
<td>Summary 72</td>
</tr>
<tr>
<td>5</td>
<td>Feed-Forward Neural Networks 73</td>
</tr>
<tr>
<td>5.1</td>
<td>Architecture of Feed-Forward Neural Networks 73</td>
</tr>
<tr>
<td>5.2</td>
<td>Learning Algorithm for Neural Networks 77</td>
</tr>
<tr>
<td>5.3</td>
<td>The Equations of Backpropagation 79</td>
</tr>
<tr>
<td>5.4</td>
<td>Drawbacks of Neural Networks (So Far) 85</td>
</tr>
<tr>
<td>5.5</td>
<td>Historical Background 85</td>
</tr>
<tr>
<td>5.6</td>
<td>References and Further Readings 86</td>
</tr>
<tr>
<td>5.7</td>
<td>Summary 86</td>
</tr>
<tr>
<td>6</td>
<td>Best Practices in Deep Learning 87</td>
</tr>
<tr>
<td>6.1</td>
<td>Minibatching 87</td>
</tr>
<tr>
<td>6.2</td>
<td>Other Optimization Algorithms 91</td>
</tr>
<tr>
<td>6.3</td>
<td>Other Activation Functions 94</td>
</tr>
<tr>
<td>6.4</td>
<td>Cost Functions 97</td>
</tr>
<tr>
<td>6.5</td>
<td>Regularization 99</td>
</tr>
<tr>
<td>6.6</td>
<td>Dropout 101</td>
</tr>
<tr>
<td>6.7</td>
<td>Temporal Averaging 102</td>
</tr>
<tr>
<td>6.8</td>
<td>Parameter Initialization and Normalization 103</td>
</tr>
<tr>
<td>6.9</td>
<td>References and Further Readings 105</td>
</tr>
<tr>
<td>6.10</td>
<td>Summary 106</td>
</tr>
<tr>
<td>7</td>
<td>Implementing Text Classification with Feed-Forward Networks 107</td>
</tr>
<tr>
<td>7.1</td>
<td>Data 108</td>
</tr>
<tr>
<td>7.2</td>
<td>Fully Connected Neural Network 109</td>
</tr>
<tr>
<td>7.3</td>
<td>Training 111</td>
</tr>
<tr>
<td>7.4</td>
<td>Summary 115</td>
</tr>
<tr>
<td>8</td>
<td>Distributional Hypothesis and Representation Learning 117</td>
</tr>
<tr>
<td>8.1</td>
<td>Traditional Distributional Representations 117</td>
</tr>
</tbody>
</table>
## Contents

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>8.2</td>
<td>Matrix Decompositions and Low-Rank Approximations</td>
<td>120</td>
</tr>
<tr>
<td>8.3</td>
<td>Drawbacks of Representation Learning Using Low-Rank Approximation</td>
<td>123</td>
</tr>
<tr>
<td>8.4</td>
<td>The Word2vec Algorithm</td>
<td>123</td>
</tr>
<tr>
<td>8.5</td>
<td>Drawbacks of the Word2vec Algorithm</td>
<td>128</td>
</tr>
<tr>
<td>8.6</td>
<td>Historical Background</td>
<td>129</td>
</tr>
<tr>
<td>8.7</td>
<td>References and Further Readings</td>
<td>130</td>
</tr>
<tr>
<td>8.8</td>
<td>Summary</td>
<td>131</td>
</tr>
<tr>
<td>9</td>
<td>Implementing Text Classification Using Word Embeddings</td>
<td>132</td>
</tr>
<tr>
<td>9.1</td>
<td>Pretrained Word Embeddings</td>
<td>132</td>
</tr>
<tr>
<td>9.2</td>
<td>Text Classification with Pretrained Word Embeddings</td>
<td>140</td>
</tr>
<tr>
<td>9.3</td>
<td>Summary</td>
<td>146</td>
</tr>
<tr>
<td>10</td>
<td>Recurrent Neural Networks</td>
<td>147</td>
</tr>
<tr>
<td>10.1</td>
<td>Vanilla Recurrent Neural Networks</td>
<td>148</td>
</tr>
<tr>
<td>10.2</td>
<td>Deep Recurrent Neural Networks</td>
<td>150</td>
</tr>
<tr>
<td>10.3</td>
<td>The Problem with Simple Recurrent Neural Networks: Vanishing Gradient</td>
<td>151</td>
</tr>
<tr>
<td>10.4</td>
<td>Long Short-Term Memory Networks</td>
<td>152</td>
</tr>
<tr>
<td>10.5</td>
<td>Conditional Random Fields</td>
<td>155</td>
</tr>
<tr>
<td>10.6</td>
<td>Drawbacks of Recurrent Neural Networks</td>
<td>163</td>
</tr>
<tr>
<td>10.7</td>
<td>Historical Background</td>
<td>163</td>
</tr>
<tr>
<td>10.8</td>
<td>References and Further Readings</td>
<td>164</td>
</tr>
<tr>
<td>10.9</td>
<td>Summary</td>
<td>164</td>
</tr>
<tr>
<td>11</td>
<td>Implementing Part-of-Speech Tagging Using Recurrent Neural Networks</td>
<td>165</td>
</tr>
<tr>
<td>11.1</td>
<td>Part-of-Speech Tagging</td>
<td>165</td>
</tr>
<tr>
<td>11.2</td>
<td>Summary</td>
<td>177</td>
</tr>
<tr>
<td>12</td>
<td>Contextualized Embeddings and Transformer Networks</td>
<td>178</td>
</tr>
<tr>
<td>12.1</td>
<td>Architecture of a Transformer Layer</td>
<td>179</td>
</tr>
<tr>
<td>12.2</td>
<td>Subword Tokenization</td>
<td>186</td>
</tr>
<tr>
<td>12.3</td>
<td>Training a Transformer Network</td>
<td>188</td>
</tr>
<tr>
<td>12.4</td>
<td>Drawbacks of Transformer Networks</td>
<td>190</td>
</tr>
<tr>
<td>12.5</td>
<td>Historical Background</td>
<td>191</td>
</tr>
<tr>
<td>12.6</td>
<td>References and Further Readings</td>
<td>192</td>
</tr>
<tr>
<td>12.7</td>
<td>Summary</td>
<td>193</td>
</tr>
</tbody>
</table>
viii Contents

13 Using Transformers with the Hugging Face Library 194
  13.1 Tokenization 194
  13.2 Text Classification 196
  13.3 Part-of-Speech Tagging 204
  13.4 Summary 215

14 Encoder-Decoder Methods 216
  14.1 BLEU: An Evaluation Measure for Machine Translation 217
  14.2 A First Sequence-to-Sequence Architecture 219
  14.3 Sequence-to-Sequence with Attention 221
  14.4 Transformer-Based Encoder-Decoder Architectures 224
  14.5 Drawbacks of Encoder-Decoder Methods 227
  14.6 Historical Background 227
  14.7 References and Further Readings 227
  14.8 Summary 228

15 Implementing Encoder-Decoder Methods 229
  15.1 Translating English to Romanian 229
  15.2 Implementation of Greedy Generation 235
  15.3 Fine-Tuning Romanian to English Translation 237
  15.4 Using a Previously Saved Model 244
  15.5 Summary 244

16 Neural Architectures for Natural Language Processing Applications 246
  16.1 Text Classification 246
  16.2 Part-of-Speech Tagging 248
  16.3 Named Entity Recognition 252
  16.4 Dependency Parsing 255
  16.5 Relation Extraction 260
  16.6 Question Answering 264
  16.7 Machine Translation 269
  16.8 Summary 271

Appendix A Overview of the Python Language and Key Libraries 272
  A.1 Python 272
  A.2 NumPy 286
  A.3 PyTorch 296
Figures


2.2 The butterfly tries to help the little monkey find her mom, but fails initially (Donaldson and Scheffler, 2008)

2.3 A depiction of a biological neuron, which captures input stimuli through its dendrites and produces an activation along its axon and synaptic terminals (left), and its computational simplification, the perceptron (right)

2.4 Visualization of the perceptron learning algorithm: (a) incorrect classification of the vector \( \mathbf{x} \) with the label Yes, for a given weight vector \( \mathbf{w} \); and (b) \( \mathbf{x} \) lies on the correct side of the decision boundary after \( \mathbf{x} \) is added to \( \mathbf{w} \)

2.5 An example of a binary classification task and a voting perceptron that aggregates two imperfect perceptrons. The voting algorithm classifies correctly all the data points by requiring two votes for the \( \times \) class to yield a \( \times \) decision. The decision boundary of the voting perceptron is shown with a continuous line

2.6 Visualization of the XOR function operating over two variables, \( x \) and \( y \). The dark circles indicate that the XOR output is 1; the clear circles stand for 0

3.1 The logistic function

3.2 Plot of the function \( f(x) = (x + 1)^2 + 1 \)

3.3 Plot of the function \( f(x) = x \sin(x)^2 + 1 \)

3.4 Multiclass logistic regression

3.5 Example of a two-dimensional decision boundary for a four-class logistic regression classifier

5.1 Decision boundary of a nonlinear classifier

5.2 Fully connected feed-forward neural network architecture. The dashed lines indicate optional components.
List of Figures xi

5.3 A feed-forward neural network with linear activation functions is a linear classifier 76
5.4 Visual helper for Equation 5.5 81
5.5 Visualization of the vanishing gradient problem for the logistic function: changes in $x$ yield smaller and smaller changes in $y$ at the two ends of the function, which means that $\frac{d}{dx}\sigma$ approaches zero in the two extremes 84
6.1 Illustration of momentum: sled 1 is more likely to get stuck in the ravine than sled 2, which starts farther up the hill, and carries momentum when it enters the ravine 92
6.2 Comparison of the tanh (continuous line) and logistic (dashed line) functions. The derivative of the tanh is larger than the derivative of the logistic for input values around zero 95
6.3 The ReLU (a) and Leaky ReLU (b) activation functions 95
6.4 A simple neural network (a), and two views of it after dropout is applied (b and c). Greyed-out nodes and edges are dropped out and thus ignored during the corresponding forward pass and backpropagation in (b) and (c) 101
8.1 Summary of the four matrices in the singular value decomposition equation: $C = UV^T$. The empty rectangles with dashed lines indicate which elements are zeroed out under the low-rank approximation 121
8.2 An illustration of the word2vec algorithm, the skip-gram variant, for the word bagel in the text: A bagel and cream cheese (also known as bagel with cream cheese) is a common food pairing in American cuisine. The algorithm clusters together output vectors for the words in the given context window (e.g., cream and cheese) with the corresponding input vector (bagel), and pushes away output vectors for words that do not appear in its proximity (e.g., computer and cat) 124
8.3 Two-dimensional projection of 1,000-dimensional vectors learned by word2vec for countries and their capitals (Mikolov et al., 2013a) 129
9.1 GloVe embedding corresponding to the word house, found in the GloVe file glove.6B.50d.txt. We have broken the vector in several lines for display purposes, but this is a single line in the text file 133
10.1 “Vanilla” recurrent neural network, where $s_i$ are state vectors, $x_i$ are input vectors, and $y_i$ are output vectors. $R$ and $O$ are functions that compute the next state and the current output vector, respectively 148
10.2 Stacked or “deep” recurrent neural network 150
10.3 Bidirectional recurrent neural network 151
List of Figures

10.4 Intuition behind the long short-term memory architecture 152
10.5 Example of a binary gate (left) and gate with real-valued elements (right) 153
10.6 Conditional random fields architecture on top of a recurrent neural network 155
10.7 Lattice of possible tag assignments for the example sentence from Figure 10.6. For simplicity, we show only four of the possible POS tags: DET – determiner, NOUN – common noun (either singular or plural), VERB – verb (any tense), and ADJ – adjective. The thick lines indicate the correct path in the lattice; the dashed lines indicate the incorrect path suggested by the first interpretation of the garden-path sentence 156
10.8 A simple lattice for the walkthrough example of the forward algorithm 160
12.1 Intuition behind transformer networks: Each output embedding is a weighted average of all input embeddings in the context 179
12.2 A transformer network consists of multiple layers, where each layer performs a weighted average of its input embeddings 180
12.3 Architecture of an individual transformer layer 181
12.4 Input example for the next sentence prediction pretraining task. [SEP] is a special separator token used to indicate end of sentence. The [CLS] token stands for class, and is used to train the binary classifier, which indicates whether sentence B follows sentence A in text. The ## marker indicates that the corresponding token is a subword token that should be appended to the token to its left 189
13.1 Confusion matrix corresponding to the long short-term memory–based part-of-speech tagger developed in Chapter 11 213
13.2 Confusion matrix corresponding to the transformer-based part-of-speech tagger 214
14.1 An encoder-decoder example of machine translation from English to Romanian, where both encoder and decoder are implemented using recurrent neural networks. Two virtual tokens, </s> and <s>, indicate end of sentence and beginning of sentence, respectively. The decoder uses the representation generated for the entire input sequence – that is, the hidden state vector e of the </s> token in the English sentence – to generate the equivalent Romanian words 217
14.2 The Vauquois triangle that describes the hierarchy of machine translation approaches 217
the input word embeddings, \( x_1 \) to \( x_n \), to produce a sequence of hidden states, \( h_1 \) to \( h_n \). The decoder is a left-to-right recurrent neural network. To avoid confusion between the source and target languages, we use \( y_t \) to indicate the input representation of the target word decoded at position \( t \), and \( s_t \) to indicate the hidden state produced by the decoder cell at position \( t \). \( c_t \) indicates the custom encoding vector of the source text for position \( t \) in the decoder.

14.4 Example of attention weights from Bahdanau et al., 2015. The x-axis corresponds to words in the source language (English); the y-axis contains the decoder words from the target language (French). Each cell visualizes an attention weight \( \alpha \) between the corresponding words, where black indicates 0 and white indicates 1.

14.5 Architecture of an individual transformer decoder layer. The decoder layer follows closely the architecture of the encoder layer (see Figure 12.3), but it includes two new components (shown in grey in the figure): a component that implements an attention mechanism between the encoded and the decoded texts, and an additional add-and-normalize layer that normalizes the outputs of the encoder-decoder attention component.

16.1 Deep averaging network (DAN) for text classification

16.2 An acceptor bidirectional recurrent neural network for text classification

16.3 Transformer network for text classification

16.4 A bidirectional transducer recurrent neural network for sequence modeling

16.5 Transformer network transducer for sequence modeling

16.6 A sample sentence parsed with universal dependencies

16.7 Dependency parsing as sequence modeling

16.8 An example of a maximum spanning tree for a hypothetical graph containing two head predictions for each sentence word. Each edge shows a (hypothetical) prediction score; these scores are included to emphasize that the maximum spanning tree has the highest overall score of all possible spanning trees.

16.9 Examples of relation mentions from the TACRED corpus, from https://nlp.stanford.edu/projects/tacred. The first example is an instance of the \texttt{per:city_of_death} relation, which holds between a person and the city where this person died; the second example is a mention of the \texttt{org:founded_by} relation, which holds between an organization and the person who...
founded it. The last example is not a relation, according to the TACRED relation schema

16.10 Relation extraction architecture with mention pooling. In this example, the first entity spans two tokens, while the second entity spans one. We omit the [CLS] and [SEP] tokens for simplicity

16.11 Relation extraction architecture with entity markers

16.12 Sample passage and question-answer pairs from the SQuAD dataset

16.13 Example of an unanswerable question from the SQuAD dataset

16.14 Example of a multiple-choice question from the QASC dataset, and the necessary facts to answer it

16.15 Examples of 3 of the 18 NLP problems that T5 trains on, all of which are formulated as text-to-text transfer. The three tasks are: English-to-German translation, summarization, and question answering

16.16 Example data point for T5 pretraining
Tables

2.1 An example of a possible feature matrix $X$ (left table) and a label vector $y$ (right table) for three animals in our story: elephant, snake, and monkey

2.2 Example output of a hypothetical classifier on five evaluation examples and two labels: positive (+) and negative (−). The “Gold” column indicates the correct labels for the five texts; the “Predicted” column indicates the classifier’s predictions

2.3 Confusion matrix showing the four possible outcomes in binary classification, where + indicates the positive label and − indicates the negative label

2.4 The feature matrix $X$ (left table) and label vector $y$ (right table) for a review classification training dataset with three examples

2.5 The perceptron learning process for the dataset shown in Table 2.4, for one pass over the training data. Both $w$ and $b$ are initialized with 0s

2.6 The feature matrix $X$ (left table) and label vector $y$ (right table) for a review classification training dataset with four examples. In this example, the only feature available is the total number of positive words in a review

3.1 Rules of computation for a few functions necessary to derive the logistic regression update rules. In these formulas, $f$ and $g$ are functions, $a$ and $b$ are constants, and $x$ is a variable

3.2 Example of a confusion matrix for three-class classification. The dataset contains 1,000 data points, with 2 data points in class $C_1$, 100 in class $C_2$, and 898 in class $C_3$

4.1 Two examples of movie reviews from IMDb. The first is a positive review of the movie *Puss in Boots (1988)*. The second is a negative review of the movie *Valentine (2001)*. These reviews can be found at [www.imdb.com/review/rw0606396](http://www.imdb.com/review/rw0606396) and [www.imdb.com/review/rw0721861](http://www.imdb.com/review/rw0721861), respectively

6.1 Three cost functions commonly used in natural language processing tasks. $m$ indicates the number of data points in the training
dataset (or the minibatch, in the case of minibatch gradient descent). $y_i$ is the correct label for example $i$.

12.1 A self-attention walkthrough example for computing the contextual embedding $z_1$ for the word *bank* in the text *bank of the river*.

12.2 Two examples of natural language processing application inputs formatted for transformer networks. In the first example, the classifier on top of the [CLS] embedding should predict the Positive label; in the second case, the prediction is Entailment.

14.1 A simple example of the BLEU evaluation measure. The underlined words indicate matches between the candidate translation and the reference. The BLEU score for this candidate translation is 3/6.

14.2 The BLEU measure allows multiple reference translations. In such cases, the highest overlap is used. In this example, the BLEU score is 4/6 due to the higher overlap with the second reference translation.

14.3 Simple overlap can be abused by repeatedly generating the same word from the reference translation. BLEU prevents this by allowing each word from a reference translation to be used just once. Naive overlap would score this candidate translation 6/6; BLEU scores it 2/6.

16.1 Universal part-of-speech tags.

16.2 Example annotations for the BIO, IO, and BILOU annotation schemas, and the CoNLL named entity types. Because in the IO representation the only label prefix is I-, it sometimes is omitted completely – for example, I-PER becomes PER. We show the prefix here for clarity.

16.3 Some universal dependency types from https://universaldependencies.org/u/dep/all.html. See this URL for the complete list of dependency types.

B.1 ASCII control characters

B.2 ASCII printable characters

B.3 The four normalization forms in Unicode
Preface

Upon encountering this publication, one might ask the obvious question, “Why do we need another deep learning and natural language processing book?” Several excellent ones have been published, covering both theoretical and practical aspects of deep learning and its application to language processing. However, from our experience teaching courses on natural language processing, we argue that, despite their excellent quality, most of these books do not target their most likely readers. The intended reader of this book is one who is skilled in a domain other than machine learning and natural language processing and whose work relies, at least partially, on the automated analysis of large amounts of data, especially textual data. Such experts may include social scientists, political scientists, biomedical scientists, and even computer scientists and computational linguists with limited exposure to machine learning.

Existing deep learning and natural language processing books generally fall into two camps. The first camp focuses on the theoretical foundations of deep learning. This is certainly useful to the aforementioned readers, as one should understand the theoretical aspects of a tool before using it. However, these books tend to assume the typical background of a machine learning researcher and, as a consequence, we have often seen students who do not have this background rapidly get lost in such material. To mitigate this issue, the second type of book that exists today focuses on the machine learning practitioner — that is, on how to use deep learning software, with minimal attention paid to the theoretical aspects. We argue that focusing on practical aspects is similarly necessary but not sufficient. Considering that deep learning frameworks and libraries have become fairly complex, the chance of misusing them due to theoretical misunderstandings is high. We have commonly seen this problem in our courses too.

This book therefore aims to bridge the theoretical and practical aspects of deep learning for natural language processing. We cover the necessary theoretical background and assume minimal machine learning background from the reader. Our aim is that anyone who took introductory linear algebra and calculus courses will be able to follow the theoretical material. To address practical aspects, this book includes pseudocode for the simpler algorithms.
discussed and actual Python code for the more complicated architectures. The code should be understandable to anyone who has taken a Python programming course. After reading this book, we expect that the reader will have the necessary foundation to immediately begin building real-world, practical natural language processing systems, and to expand their knowledge by reading research publications on these topics.