

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.

MIHAI SURDEANU 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.

MARCO A. VALENZUELA-ESCÁRCEGA 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



CAMBRIDGE
UNIVERSITY PRESS

Cambridge University Press & Assessment
978-1-316-51566-2 — Deep Learning for Natural Language Processing
Mihai Surdeanu, Marco Antonio Valenzuela-Escárcega
Frontmatter
[More Information](#)



Shaftesbury Road, Cambridge CB2 8EA, United Kingdom

One Liberty Plaza, 20th Floor, New York, NY 10006, USA

477 Williamstown Road, Port Melbourne, VIC 3207, Australia

314–321, 3rd Floor, Plot 3, Splendor Forum, Jasola District Centre,
New Delhi – 110025, India

103 Penang Road, #05–06/07, Visioncrest Commercial, Singapore 238467

Cambridge University Press is part of Cambridge University Press & Assessment,
a department of the University of Cambridge.

We share the University's mission to contribute to society through the pursuit of
education, learning and research at the highest international levels of excellence.

www.cambridge.org

Information on this title: www.cambridge.org/9781316515662

DOI: 10.1017/9781009026222

© Mihai Surdeanu and Marco Antonio Valenzuela-Escárcega 2024

This publication is in copyright. Subject to statutory exception and to the provisions
of relevant collective licensing agreements, no reproduction of any part may take
place without the written permission of Cambridge University Press & Assessment.

First published 2024

A catalogue record for this publication is available from the British Library

*A Cataloging-in-Publication data record for this book is available from the Library of
Congress*

ISBN 978-1-316-51566-2 Hardback

ISBN 978-1-009-01265-2 Paperback

Cambridge University Press & Assessment has no responsibility for the persistence
or accuracy of URLs for external or third-party internet websites referred to in this
publication and does not guarantee that any content on such websites is, or will
remain, accurate or appropriate.

Contents

<i>List of Figures</i>	<i>page</i> x
<i>List of Tables</i>	xv
<i>Preface</i>	xvii
1 Introduction	1
1.1 What This Book Covers	3
1.2 What This Book Does Not Cover	5
1.3 Deep Learning Is Not Perfect	5
1.4 Mathematical Notations	6
2 The Perceptron	8
2.1 Machine Learning Is Easy	8
2.2 Use Case: Text Classification	11
2.3 Evaluation Measures for Text Classification	12
2.4 The Perceptron	14
2.5 Voting Perceptron	22
2.6 Average Perceptron	24
2.7 Drawbacks of the Perceptron	26
2.8 Historical Background	28
2.9 References and Further Readings	29
2.10 Summary	29
3 Logistic Regression	30
3.1 The Logistic Regression Decision Function and Learning Algorithm	30
3.2 The Logistic Regression Cost Function	32
3.3 Gradient Descent	34
3.4 Deriving the Logistic Regression Update Rule	38
3.5 From Binary to Multiclass Classification	40
3.6 Evaluation Measures for Multiclass Text Classification	43
	v

vi	Contents	
3.7	Drawbacks of Logistic Regression	46
3.8	Historical Background	46
3.9	References and Further Readings	47
3.10	Summary	48
4	Implementing Text Classification Using Perceptron and Logistic Regression	49
4.1	Binary Classification	49
4.2	Multiclass Classification	62
4.3	Summary	72
5	Feed-Forward Neural Networks	73
5.1	Architecture of Feed-Forward Neural Networks	73
5.2	Learning Algorithm for Neural Networks	77
5.3	The Equations of Backpropagation	79
5.4	Drawbacks of Neural Networks (So Far)	85
5.5	Historical Background	85
5.6	References and Further Readings	86
5.7	Summary	86
6	Best Practices in Deep Learning	87
6.1	Minibatching	87
6.2	Other Optimization Algorithms	91
6.3	Other Activation Functions	94
6.4	Cost Functions	97
6.5	Regularization	99
6.6	Dropout	101
6.7	Temporal Averaging	102
6.8	Parameter Initialization and Normalization	103
6.9	References and Further Readings	105
6.10	Summary	106
7	Implementing Text Classification with Feed-Forward Networks	107
7.1	Data	108
7.2	Fully Connected Neural Network	109
7.3	Training	111
7.4	Summary	115
8	Distributional Hypothesis and Representation Learning	117
8.1	Traditional Distributional Representations	117

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

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
<i>Appendix A Overview of the Python Language and Key Libraries</i>		272
A.1	Python	272
A.2	NumPy	286
A.3	PyTorch	296

Contents	ix
<i>Appendix B</i> Character Encodings: ASCII and Unicode	301
B.1 How Do Computers Represent Text?	301
B.2 How to Encode/Decode Characters in Python	304
B.3 Text Normalization	305
<i>References</i>	308
<i>Index</i>	321

Figures

2.1	A wonderful children’s book that introduces the fundamentals of machine learning: <i>Where’s My Mom?</i> , by Julia Donaldson and Axel Scheffler © Julia Donaldson 2000, illustrations copyright © Axel Scheffler 2000	page 9
2.2	The butterfly tries to help the little monkey find her mom, but fails initially (Donaldson and Scheffler, 2008)	10
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)	15
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}	19
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	23
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	26
3.1	The logistic function	31
3.2	Plot of the function $f(x) = (x + 1)^2 + 1$	35
3.3	Plot of the function $f(x) = x \sin(x)^2 + 1$	37
3.4	Multiclass logistic regression	41
3.5	Example of a two-dimensional decision boundary for a four-class logistic regression classifier	47
5.1	Decision boundary of a nonlinear classifier	74
5.2	Fully connected feed-forward neural network architecture. The dashed lines indicate optional components.	74

	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: $\mathbf{C} = \mathbf{U}\Sigma\mathbf{V}^T$. 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 <i>bagel</i> in the text: <i>A bagel and cream cheese (also known as bagel with cream cheese) is a common food pairing in American cuisine.</i> The algorithm clusters together output vectors for the words in the given context window (e.g., <i>cream</i> and <i>cheese</i>) with the corresponding input vector (<i>bagel</i>), and pushes away output vectors for words that do not appear in its proximity (e.g., <i>computer</i> and <i>cat</i>)	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 <i>house</i> , found in the GloVe file <code>glove.6B.50d.txt</code> . 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 \mathbf{s}_i are state vectors, \mathbf{x}_i are input vectors, and \mathbf{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

xii	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 <i>class</i> , 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 c 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
14.3	The architecture of a single decoder cell in a sequence-to-sequence architecture with attention. The encoder for this architecture is a bidirectional recurrent neural network that uses	

List of Figures	xiii
the input word embeddings, \mathbf{x}_1 to \mathbf{x}_n , to produce a sequence of hidden states, \mathbf{h}_1 to \mathbf{h}_n . The decoder is a left-to-right recurrent neural network. To avoid confusion between the source and target languages, we use \mathbf{y}_t to indicate the input representation of the target word decoded at position t , and \mathbf{s}_t to indicate the hidden state produced by the decoder cell at position t . \mathbf{c}_t indicates the custom encoding vector of the source text for position t in the decoder	222
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 α between the corresponding words, where black indicates 0 and white indicates 1	224
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	226
16.1 Deep averaging network (DAN) for text classification	247
16.2 An acceptor bidirectional recurrent neural network for text classification	247
16.3 Transformer network for text classification	248
16.4 A bidirectional transducer recurrent neural network for sequence modeling	251
16.5 Transformer network transducer for sequence modeling	251
16.6 A sample sentence parsed with universal dependencies	255
16.7 Dependency parsing as sequence modeling	257
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	260
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 <code>per:city_of_death</code> relation, which holds between a person and the city where this person died; the second example is a mention of the <code>org:founded_by</code> relation, which holds between an organization and the person who	

xiv	List of Figures	
	founded it. The last example is not a relation, according to the TACRED relation schema	261
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	263
16.11	Relation extraction architecture with entity markers	264
16.12	Sample passage and question-answer pairs from the SQuAD dataset	265
16.13	Example of an unanswerable question from the SQuAD dataset	265
16.14	Example of a multiple-choice question from the QASC dataset, and the necessary facts to answer it	267
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	270
16.16	Example data point for T5 pretraining	270

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	page 10
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	12
2.3	Confusion matrix showing the four possible outcomes in binary classification, where + indicates the positive label and − indicates the negative label	13
2.4	The feature matrix X (left table) and label vector y (right table) for a review classification training dataset with three examples	20
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	21
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 <i>total</i> number of positive words in a review	22
3.1	Rules of computation for a few functions necessary to derive the logistic regression update rules. In these formulas, <i>f</i> and <i>g</i> are functions, <i>a</i> and <i>b</i> are constants, and <i>x</i> is a variable	40
3.2	Example of a confusion matrix for three-class classification. The dataset contains 1,000 data points, with 2 data points in class <i>C1</i> , 100 in class <i>C2</i> , and 898 in class <i>C3</i>	44
4.1	Two examples of movie reviews from IMDb. The first is a positive review of the movie <i>Puss in Boots</i> (1988). The second is a negative review of the movie <i>Valentine</i> (2001). These reviews can be found at www.imdb.com/review/rw0606396 and www.imdb.com/review/rw0721861 , respectively	51
6.1	Three cost functions commonly used in natural language processing tasks. <i>m</i> indicates the number of data points in the training	

xvi	List of Tables	
	dataset (or the minibatch, in the case of minibatch gradient descent). y_i is the correct label for example i	97
12.1	A self-attention walkthrough example for computing the contextual embedding z_1 for the word <i>bank</i> in the text <i>bank of the river</i>	184
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	191
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	218
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	218
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	219
16.1	Universal part-of-speech tags	250
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	254
16.3	Some universal dependency types from https://universaldependencies.org/u/dep/all.html . See this URL for the complete list of dependency types	256
B.1	ASCII control characters	302
B.2	ASCII printable characters	303
B.3	The four normalization forms in Unicode	306

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

xviii Preface

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.