1 Introduction

Machine learning (ML) has become a pervasive part of our lives. For example, Pedro Domingos, an ML faculty member at the University of Washington, discusses a typical day in the life of a twenty-first-century person, showing how she is accompanied by ML applications throughout the day from early in the morning (e.g., waking up to music that the machine matched to their preferences) to late at night (e.g., taking a drug designed by a biomedical researcher with the help of a robot scientist) (Domingos, 2015). Of all approaches in ML, *deep learning* has seen explosive success in the past decade, and today it is ubiquitous in real-world applications of ML. At a high level, deep learning is a subfield of ML that focuses on artificial neural networks, which were "inspired by information processing and distributed communication nodes in biological systems."[1]

Natural language processing (NLP) is an important interdisciplinary field that lies at the intersection of linguistics, computer science, and ML. In general, NLP deals with programming computers to process and analyze large amounts of natural language data.[2] As an example of its usefulness, consider that PubMed, a repository of biomedical publications built by the National Institutes of Health, has indexed more than one million research publications *per year* since 2010 (Vardakas et al., 2015).[3] Clearly, no human reader (or team of readers) can process so much material. We need machines to help us manage this vast amount of knowledge. As one example out of many, an interdisciplinary collaboration that included our research team showed that machine reading discovers an order of magnitude more protein signaling pathways in biomedical literature than exist today in humanly curated knowledge bases (Valenzuela-Escárcega et al., 2018).[4] Only 60–80% of these automatically discovered biomedical interactions are correct (a good motivation for *not* letting the machines work alone!). But, without NLP, all of these would

[1] `https://en.wikipedia.org/wiki/Deep_learning`.

[2] `https://en.wikipedia.org/wiki/Natural_language_processing`.

[3] `www.ncbi.nlm.nih.gov/pubmed`.

[4] Protein signaling pathways "govern basic activities of cells and coordinate multiple-cell actions." Errors in these pathways "may cause diseases such as cancer." See `https://en.wikipedia.org/wiki/Cell_signaling`.

https://doi.org/10.1017/9781009026222.002 Published online by Cambridge University Press

remain "undiscovered public knowledge" (Swanson, 1986), limiting our ability to understand important diseases such as cancer. Other important and more common applications of NLP include web search, machine translation, and speech recognition, all of which have had a major impact in almost everyone's life.

Since approximately 2014, a "deep learning tsunami" has hit the field of NLP (Manning, 2015) to the point that, today, a majority of NLP publications use deep learning. For example, the percentage of deep learning publications at four top NLP conferences increased from under 40% in 2012 to 70% in 2017 (Young et al., 2018). There is good reason for this domination: deep learning systems are relatively easy to build (due to their modularity), and they perform better than many other ML methods.[5] For example, the site nlpprogress.com, which keeps track of state-of-the-art results in many NLP tasks, is dominated by results of deep learning approaches.

This book explains deep learning methods for NLP, aiming to cover both theoretical aspects (e.g., how do neural networks learn?) and practical ones (e.g., how do I build one for language applications?).

The goal of the book is to do this while assuming minimal technical background from the reader. The theoretical material in the book should be completely accessible to the reader who took linear algebra, calculus, and introduction to probability theory courses, or who is willing to do some independent work to catch up. From linear algebra, the most complicated notion used is matrix multiplication. From calculus, we use differentiation and partial differentiation. From probability theory, we use conditional probabilities and independent events. The code examples should be understandable to the reader who took a Python programming course.

Starting nearly from scratch aims to address the background of who we think will be the typical reader of this book: an expert in a discipline other than ML and NLP, but who needs ML and NLP for her job. There are many examples of such disciplines: the social scientist who needs to mine social media data, the political scientist who needs to process transcripts of political discourse, the business analyst who has to parse company financial reports at scale, the biomedical researcher who needs to extract cell signaling mechanisms from publications, and so forth. Further, we hope this book will also be useful to computer scientists and computational linguists who need to catch up with the deep learning wave. In general, this book aims to mitigate the impostor syndrome (Dickerson, 2019) that affects many of us in this era of rapid change in the field of ML and artificial intelligence (AI) (this author certainly has suffered and still suffers from it![6]).

[5] However, they are not perfect. See Section 1.3 for a discussion.

[6] Even the best of us suffer from it. Please see Kevin Knight's description of his personal experience involving tears (not of joy) in the introduction of this tutorial (Knight, 2009).

1.1 What This Book Covers

This book interleaves chapters that discuss the theoretical aspects of deep learning for NLP with chapters that focus on implementing the previously discussed theory. For the implementation chapters, we will use PyTorch, a deep learning library that is well suited for NLP applications.[7]

Chapter 2 begins the theory thread of the book by attempting to convince the reader that ML is easy. We use a children's book to introduce key ML concepts, including our first learning algorithm. From this example, we start building several basic neural networks. In the same chapter, we formalize the perceptron algorithm, the simplest neural network. In Chapter 3, we transform the perceptron into a logistic regression network, another simple neural network that is surprisingly effective for NLP. In Chapters 5 and 6, we generalize these algorithms into feed-forward neural networks, which operate over arbitrary combinations of artificial neurons.

The astute historian of deep learning will have observed that deep learning had an impact earlier on image processing than on NLP. For example, in 2012, researchers at the University of Toronto reported a massive improvement in image classification when using deep learning (Krizhevsky et al., 2012). However, it took more than two years to observe similar performance improvements in NLP. One explanation for this delay is that image processing starts from very low-level units of information (i.e., the pixels in the image), which are then hierarchically assembled into blocks that are more and more semantically meaningful (e.g., lines and circles, then eyes and ears, in the case of facial recognition). In contrast, NLP starts from words, which are packed with a lot more semantic information than pixels and, because of that, are harder to learn from. For example, the word *house* packs a lot of commonsense knowledge (e.g., houses generally have windows and doors and they provide shelter). Although this information is shared with other words (e.g., *building*), a learning algorithm that has seen *house* in its training data will not know how to handle the word *building* in a new text to which it is exposed after training.

Chapter 8 addresses this limitation. In it, we discuss word2vec, a method that transforms words into a numerical representation that captures (some) semantic knowledge. This technique is based on the observation that "you shall know a word by the company it keeps" (Firth, 1957) – that is, it learns these semantic representations from the context in which words appear in large collections of texts. Under this formalization, similar words such as *house* and *building* will have similar representations, which will improve the learning capability of our neural networks. An important limitation of word2vec is that it conflates all senses of a given word into a single numerical representation. That is, the word *bank* gets a single numerical representation regardless of whether its

[7] `https://pytorch.org`.

https://doi.org/10.1017/9781009026222.002 Published online by Cambridge University Press

current context indicates a financial sense – for example, *Bank of London* – or a geological one – for example, *bank of the river*.

Chapter 10 introduces sequence models for processing text. For example, while the word *book* is syntactically ambiguous (i.e., it can be either a noun or a verb), the information that it is preceded by the determiner *the* in a text gives strong hints that this instance of it is a noun. In this chapter, we cover recurrent neural network architectures designed to model such sequences, including long short-term memory networks and conditional random fields.

This word2vec limitation is addressed in Chapter 12 with contextualized embeddings that are sensitive to a word's surroundings. These contextualized embeddings are built using transformer networks that rely on "attention," a mechanism that computes the representation of a word using a weighted average of the representations of the words in its context. These weights are learned and indicate how much "attention" each word should pay to each of its neighbors (hence the name).

Chapter 14 discusses encoder-decoder methods (i.e., methods tailored for NLP tasks that require the transformation of one text into another). The most common example of such a task is machine translation, for which the input is a sequence of words in one language, and the output is a sequence that captures the translation of the original text in a new language.

Chapter 16 shows how several NLP applications such as part-of-speech tagging, syntactic parsing, relation extraction, question answering, and machine translation can be robustly implemented using the neural architectures introduced previously.

As mentioned before, the theoretical discussion in these chapters is interleaved with chapters that discuss how to implement these notions in PyTorch. Chapter 4 shows an implementation of the perceptron and logistic regression algorithms introduced in Chapters 2 and 3 for a text classification application. Chapter 7 presents an implementation of the feed-forward neural network introduced in Chapters 5 and 6 for the same application. Chapter 9 enhances the previous implementation of a neural network with the continuous word representations introduced in Chapter 8.

Chapter 11 implements a part-of-speech tagger using the recurrent neural networks introduced in Chapter 10. Chapter 13 shows the implementation of a similar part-of-speech tagger using the contextualized embeddings generated by a transformer network. The same chapter also shows how to use transformer networks for text classification. Last, Chapter 15 implements a machine translation application using some of the encoder-decoder methods discussed in Chapter 14.

We recommend that the reader not familiar with the Python programming language first read Appendixes A and B for a brief overview of the programming language and pointers on how to handle international characters represented in Unicode in Python.

1.2 What This Book Does Not Cover

It is important to note that deep learning is only one of the many subfields of ML. In his book, Domingos provides an intuitive organization of these subfields into five "tribes" (Domingos, 2015):

Connectionists: This tribe focuses on ML methods that (shallowly) mimic the structure of the brain. The methods described in this book fall into this tribe.

Evolutionaries: The learning algorithms adopted by this group of approaches, also known as genetic algorithms, focus on the "survival of the fittest." That is, these algorithms "mutate" the "DNA" (or parameters) of the models to be learned, and preserve the generations that perform the best.

Symbolists: The symbolists rely on inducing logic rules that explain the data in the task at hand. For example, a part-of-speech tagging system in this camp may learn a rule such as `if` the previous word is *the*, `then` the next word is a noun.

Bayesians: The Bayesians use probabilistic models such as Bayesian networks. All these methods are driven by Bayes's rule, which describes the probability of an event.

Analogizers: The analogizers' methods are motivated by the observation that "you are what you resemble." For example, a new email is classified as spam because it uses content similar to other emails previously classified as such.

It is beyond the goal of this book to explain these other tribes in detail. For a more general description of ML, the interested reader should look to other sources such as Domingos's book or Hal Daumé III's excellent *Course in Machine Learning*.[8]

Even from the connectionist tribe, we focus only on neural methods that are relevant for fundamental language processing and that we hope serve as a solid stepping stone toward research in NLP.[9] Other important, more advanced topics are not discussed. These include: domain adaptation, reinforcement learning, dialog systems, and methods that process multimodal data such as text and images.

1.3 Deep Learning Is Not Perfect

While deep learning has pushed the performance of many ML applications beyond what we thought possible just 10 years ago, it is certainly not perfect. Gary Marcus and Ernest Davis provide a thoughtful criticism of deep learning in their book, *Rebooting AI* (Marcus and Davis, 2019). Their key arguments are:

[8] `http://ciml.info`.

[9] Most methods discussed in this book are certainly useful and commonly used outside of NLP as well.

https://doi.org/10.1017/9781009026222.002 Published online by Cambridge University Press

Deep learning is opaque: While deep learning methods often learn well, it is unclear *what* is learned – that is, what the connections between the network neurons encode. This is dangerous, as biases and bugs may exist in the models learned, and they may be discovered only too late, when these systems are deployed in important real-world applications such as medical diagnoses or self-driving cars.

Deep learning is brittle: It has been repeatedly shown both in the ML literature and in actual applications that deep learning systems (and for that matter most other ML approaches) have difficulty adapting to new scenarios they have not seen during training. For example, self-driving cars that were trained in regular traffic on US highways or large streets do not know how to react to unexpected situation such as a firetruck stopped on a highway.[10]

Deep learning has no common sense: An illustrative example of this limitation is that object recognition classifiers based on deep learning tend to confuse objects when they are rotated in three-dimensional space – for example, an overturned bus in the snow is confused with a snowplow. This happens because deep learning systems lack the commonsense knowledge that some object features are inherent properties of the category itself regardless of the object position – for example, a school bus in the United States usually has a yellow roof, while some features are just contingent associations – for example, snow tends to be present around snowplows. (Most) humans naturally use common sense, which means that we do generalize better to novel instances, especially when they are outliers.

All the issues Marcus and Davis raised remain largely unsolved today.

1.4 Mathematical Notations

While we try to rely on plain language as much as possible in this book, mathematical formalisms cannot (and should not) be avoided. Where mathematical notations are necessary, we rely on the following conventions:

- We use lowercase characters such as x to represent scalar values, which will generally have integer or real values.
- We use bold lowercase characters such as $\mathbf{x}$ to represent arrays (or vectors) of scalar values, and x_i to indicate the scalar element at position i in this vector. Unless specified otherwise, we consider all vectors to be column vectors during operations such as multiplication, even though we show them in text as horizontal. We use $[\mathbf{x}; \mathbf{y}]$ to indicate vector concatenation. For example, if $\mathbf{x} = (1, 2)$ and $\mathbf{y} = (3, 4)$, then $[\mathbf{x}; \mathbf{y}] = (1, 2, 3, 4)$.
- We use bold uppercase characters such as $\mathbf{X}$ to indicate matrices of scalar values. Similarly, x_{ij} points to the scalar element in the matrix at row i

[10] `www.teslarati.com/tesla-model-s-firetruck-crash-details`.

https://doi.org/10.1017/9781009026222.002 Published online by Cambridge University Press

and column j. $\mathbf{x}_i$ indicates the vector corresponding to the entire row i in matrix $\mathbf{X}$.

- We collectively refer to matrices of arbitrary dimensions as *tensors*. By and large, in this book, tensors will have one dimension (i.e. vectors) or two (matrices). Occasionally, we will run into tensors with three dimensions.
- A word with an arrow on top refers to the *distributional representation* or *embedding vector* corresponding to that word. For example, $\vec{queen}$ indicates the embedding vector for the word *queen*.

https://doi.org/10.1017/9781009026222.002 Published online by Cambridge University Press