Содержание

Об авторе

Тушан Ганегедара написал эту книгу на третьем году обучения в аспирантуре университета Сиднея, Австралия, а перед этим получил степень бакалавра с отличием в университете Моратува, Шри-Ланка. Он специализируется на машинном обучении и особенно увлечен глубокими нейросетями. Тушан любит рисковать и часто запускает алгоритмы на непроверенных данных. Он также работает в качестве главного аналитика данных в австралийском стартапе AssessThreat и регулярно пишет технические статьи и учебные пособия по машинному обучению. Кроме того, он стремится к здоровому образу жизни и ежедневно занимается плаванием.

Я хочу поблагодарить моих родителей, моих братьев и сестер и мою жену за веру в меня и за поддержку, которую они оказали, а также всех моих учителей и моего научного руководителя за наставления, которые они дали мне.

О рецензентах

Мотаз Саад окончил аспирантуру по информатике в Университете Лотарингии. Он любит данные и все, что с ними связано. Он более 10 лет занимается обработкой естественных языков, компьютерной лингвистикой, наукой о данных и машинным обучением. В настоящее время работает доцентом на факультете информационных технологий IUG (Islamic University of Gaza).

Доктор Джозеф О'Коннор – специалист по данным, искренне увлеченный глубоким обучением. Его компания Deep Learn Analytics, британская консалтинговая компания, специализирующаяся на данных, оказывает предприятиям услуги по разработке приложений и инфраструктуры машинного обучения от концепции до развертывания. Ему была присуждена степень доктора философии в университете Лондона за работу по анализу данных эксперимента по физике высоких энергий MINOS. С того времени он разработал продукты машинного обучения для ряда компаний частного сектора, специализирующихся на NLP и прогнозировании временных рядов. Вы можете найти его на **http://deeplearnanalytics.com/**.

Предисловие

В наш век цифровой информации, в котором мы живем, объем данных растет в геометрической прогрессии. Пока вы читаете эти слова, мировые запасы данных продолжают расти с ошеломляющей скоростью. Большая часть этих данных относится к языковым данным – текстовым или устным, – таким как электронные письма, сообщения в социальных сетях, телефонные звонки и статьи в интернете. Машинная *обработка естественного языка* (natural language processing, NLP) эффективно использует эти данные, чтобы помочь людям в их бизнесе или в повседневных задачах. Технология NLP уже произвела революцию в повседневном использовании данных, помогает бизнесу, облегчает жизнь людей и продолжит делать это в будущем.

Одним из наиболее распространенных примеров применения NLP являются *виртуальные помощники* (virtual assistants, VA), такие как Siri от Apple, Google Assistant и Amazon Alexa. Всякий раз, когда вы просите своего виртуального помощника найти «отели в Швейцарии по низким ценам», запускается серия сложных задач обработки естественного языка. Во-первых, ваш помощник должен понять смысл запроса (например, определить, что надо искать низкие цены на отели, а не ближайшие площадки для выгула собак). Еще одно решение, которое должен принять помощник, – это определить критерий «низкой цены». Затем помощник должен ранжировать города в Швейцарии, возможно, исходя из вашей прошлой истории путешествий. Помощник может просканировать сайты агрегаторов отелей, чтобы извлечь оттуда цены на отели в Швейцарии и «прочитать» отзывы посетителей для каждого отеля. Как видите, ответ на запрос, который вы получаете через несколько секунд, является результатом разносторонней работы системы NLP.

Итак, что делает системы NLP настолько универсальными и точными в решении наших повседневных задач? В основе успеха лежат алгоритмы глубокого обучения. Это, по сути, сложные нейронные сети, которые могут проецировать необработанные данные в желаемый результат, не требуя какой-либо сложной ручной настройки алгоритма. Это означает, что турист напишет отзыв об отеле на естественном человеческом языке, а компьютер безошибочно ответит на вопрос «Насколько положителен отзыв клиента об этом отеле?». Кроме того, глубокое обучение уже достигло и даже превысило уровень человеческих возможностей в различных задачах NLP, таких как распознавание речи и машинный перевод.

Прочитав эту книгу, вы узнаете, как решать многие интересные задачи NLP, используя глубокое обучение. Итак, если вы хотите быть влиятельным человеком, который меняет мир, изучение NLP имеет решающее значение. Прикладные задачи варьируются от изучения семантики слов до генерации новых историй и выполнения машинного перевода с обучением «на ходу». Все главы содержат упражнения, практические примеры и пошаговые инструкции по внедрению рассматриваемых решений. Для всех упражнений в этой книге мы будем использовать Python с TensorFlow – популярной библиотекой распределенных вычислений, которая делает реализацию глубоких нейронных сетей очень простой и удобной.

Для кого эта книга

Эта книга предназначена для начинающих исследователей и разработчиков, которые стремятся сделать мир лучше, используя лингвистические данные. Книга предоставит вам прочную практическую основу для решения задач NLP. В этой книге мы рассмотрим различные аспекты NLP, уделяя больше внимания практической реализации, чем теоретическим основам. Обладание практическими навыками решения различных задач NLP поможет вам совершить более плавный переход к изучению более сложных теоретических аспектов этих методов. Кроме того, хорошее понимание практической части NLP поможет при выполнении более точной настройки ваших алгоритмов, чтобы получить максимальную отдачу от конкретного проекта.

Какие темы охватывает эта книга

Глава 1 знакомит вас с NLP. В этой главе вы узнаете причины, по которым востребована обработка естественного языка. Далее перечислены некоторые общие проблемы и определены две основные эпохи NLP – период использования традиционных методов и современные приемы глубокого обучения. Сначала мы в общих чертах рассмотрим, как задача моделирования языка решается с помощью традиционных алгоритмов. Затем обсудим современную эпоху, когда задачи машинной обработки языка решаются с помощью моделей глубокого обучения, и рассмотрим основные семейства алгоритмов глубокого обучения. Потом вы познакомитесь с принципом работы одного из основных современных алгоритмов – полносвязной нейронной сети. Главу завершает «дорожная карта», в которой дается краткое введение в последующие главы.

Глава 2 знакомит вас с библиотекой Python TensorFlow – основной платформой, на которой реализованы все решения, рассмотренные в этой книге. Вы начнете с написания кода для реализации простых вычислений в TensorFlow, а затем узнаете, как выполняется этот код, начиная с запуска программы и заканчивая получением результатов. Таким образом, вы на простом примере познакомитесь с основными компонентами TensorFlow. Вы еще больше укрепите понимание TensorFlow благодаря красочной аналогии с процессом выполнения заказов в ресторане. Далее мы обсудим технические детали TensorFlow, такие как структуры данных и операции с нейронными сетями, встроенные в TensorFlow. Наконец, вы создадите полносвязную нейронную сеть для распознавания рукописных цифр. Это пример реализации комплексного решения с помощью TensorFlow.

Глава 3 начинается с обсуждения того, как решать задачи NLP с помощью TensorFlow. В этой главе вы узнаете, как нейронные сети применяются для изучения векторов слов, известных также как представления слов. Векторы слов являют собой числовые представления слов с учетом смыслового окружения. Сначала мы обсудим несколько традиционных подходов к достижению этой цели, которые включают использование большой базы знаний, созданной человеком, известной как WordNet. Затем мы рассмотрим современный подход на основе нейронных сетей, известный как Word2vec и основанный на изучении векторов слов без вмешательства человека. Сначала вы исследуете механику Word2vec, проработав практиче-

ский пример, а затем познакомитесь с двумя усовершенствованными методами – *словосочетаниями с пропуском* (Skip-Gram) и *непрерывным мультимножеством слов* (continuous bag-of-words, CBOW). Главу завершает обсуждение концептуальных особенностей алгоритмов, а также способы их реализации в TensorFlow.

Глава 4 раскрывает более сложные темы, связанные с векторами слов. Сначала мы сравним подходы Skip-Gram и CBOW, чтобы узнать, существует ли победитель. Далее мы обсудим несколько улучшений, которые можно использовать для повышения качества работы алгоритмов Word2vec. Затем вы познакомитесь с более современным и мощным подходом к изучению векторизации смыслов – алгоритмом *глобальных векторов* GloVe. Наконец, вы познакомитесь с векторами слов в действии в задаче классификации документов. В этом упражнении вы увидите, что векторный подход достаточно точен и может правильно классифицировать тему документа.

Глава 5 посвящена обсуждению *сверточных нейронных сетей* (convolutional neural network, CNN) – семейства нейронных сетей, которые превосходны при обработке пространственных данных, таких как изображения или предложения. Мы начнем с изучения общих принципов работы CNN, обсудив, как они обрабатывают данные и какие операции выполняют. Далее мы детально разберем каждую операцию, связанную с вычислениями, чтобы понять внутреннюю математику CNN. Наконец, вы выполните два упражнения. Во-первых, вы будете классифицировать изображения рукописных цифр с помощью CNN. Вы убедитесь, что CNN способны быстро достичь очень высокой точности при решении задач такого типа. Далее вы узнаете, как можно использовать CNN для классификации предложений. В завершающем главу примере нейросеть предсказывает, относится ли предложение к объекту, человеку, местоположению и т. д.

Глава 6 рассказывает о *рекуррентных нейронных сетях* (recurrent neural network, RNN) – мощном семействе нейронных сетей, которые могут обрабатывать последовательности данных. Сначала вы изучите внутреннюю математику и правила функционирования RNN в процессе обучения. Затем вы познакомитесь с различными вариантами RNN и их применением (например, структуры *один-к-одному* и *один-ко-многим*). Наконец, вы выполните упражнение, в котором RNN решает задачу генерации текста. В этом примере RNN обучается на наборе сказок и пытается сочинить новую сказку. Вы увидите, что RNN плохо работают с долговременной памятью. Наконец, мы обсудим более продвинутый вариант RNN под названием RNN-CF, обладающий более надежной долгосрочной памятью.

Глава 7 рассказывает про более мощные рекуррентные нейросети, которые способны запоминать данные в течение более длительного периода времени, поскольку стандартные RNN плохо поддерживают долговременную память. В этой главе рассказано про сети с *долгой краткосрочной памятью* (long short-term memory, LSTM). Известно, что LSTM превосходят другие последовательные модели во многих задачах временных рядов. Сначала вы исследуете базовую математику и новые правила LSTM на ярком примере, который иллюстрирует, почему каждое вычисление имеет значение. Затем вы узнаете, как LSTM могут сохранять память еще дольше. Далее мы обсудим, как можно улучшить прогнозные возможности LSTM. Наконец, мы рассмотрим несколько вариантов LSTM, имеющих более сложную структуру, включая *управляемые рекуррентные блоки* (Gated Recurrent Unit, GRU).

Глава 8 подробно рассказывает о том, как LSTM-сети работают в задаче генерации текста. Вы качественно и количественно оцените, насколько хорош текст, сгенерированный LSTM, а также проведете сравнение различных модификаций LSTM. Наконец, вы узнаете, как добавить в модель механизм векторного представления слов, чтобы улучшить качество сгенерированного текста.

Глава 9 от работы с текстами переносит вас к *мультимодальным данным*, то есть комбинации изображения и текста. В этой главе вы узнаете, как автоматически генерировать текстовое описание для заданного изображения. Решение включает в себя объединение сверточной модели прямого распространения (CNN) со слоем представления слов и последовательной модели (LSTM) таким образом, чтобы сформировать полный цикл машинного обучения.

Глава 10 рассказывает о реализации модели *нейронного машинного перевода* (neural machine translation, NMT). В машинном переводе мы переводим предложение/фразу с исходного языка на целевой язык. Сначала вы познакомитесь с краткой историей и основами машинного перевода. Затем детально изучите архитектуру современных моделей нейронного машинного перевода, включая процедуры обучения и вывода. Далее вы узнаете, как реализовать систему NMT с нуля. Наконец, вы познакомитесь со способами улучшения стандартных систем NMT.

Глава 11 завершает книгу и посвящена текущим тенденциям и будущему NLP. Мы обсудим последние открытия, связанные с системами и задачами, о которых рассказано в предыдущих главах. В этой главе будет рассказано о самых интересных нововведениях, а также будет дано углубленное представление о внедрении некоторых технологий.

Приложение познакомит читателя с различными математическими структурами данных и операциями. Мы также обсудим несколько важных понятий в теории вероятности. Затем вы познакомитесь с Keras – библиотекой высокого уровня, которая использует TensorFlow. Keras упрощает реализацию нейронных сетей, скрывая некоторые детали в TensorFlow, что может сначала показаться запутанным. Вы познакомитесь с примером реализации сверточной сети с помощью Keras. Далее мы рассмотрим использование библиотеки seq2seq в TensorFlow для реализации системы машинного перевода с гораздо меньшим количеством кода, чем в упражнении из главы 11. Наконец, вы познакомитесь с руководством по использованию TensorBoard для визуализации смысловых связей слов. TensorBoard – это удобный инструмент визуализации, который поставляется с TensorFlow. Его можно использовать для визуализации и мониторинга различных переменных в вашем клиенте TensorFlow.

Как получить максимальную отдачу от этой книги

Чтобы извлечь максимальную пользу из этой книги, читатель должен:

- иметь твердую волю и желание изучить современные методы NLP;
- ознакомиться с базовым синтаксисом языка Python и структурами данных, такими как списки и словари;
- знать основы математики, например умножение матриц/векторов.

Не обязательно, но желательно:

- иметь достаточно обширные знания в области математики, чтобы лучше понять разделы, в которых идет речь о конкретных проблемах и методах их решения;
- прочитать научные статьи по тематике глубокого обучения, чтобы иметь представление о достижениях науки и методах, помимо тех, что описаны в книге.

Условные обозначения и соглашения, принятые в книге

В книге используются следующие типографские соглашения.

Курсив – используется для смыслового выделения важных положений, новых терминов, имен команд и утилит, а также слов и предложений на естественном языке.

`Моноширинный шрифт` – применяется для листингов программ, а также в обычном тексте для обозначения имен переменных, функций, типов, объектов, баз данных, переменных среды, операторов, ключевых слов и других программных конструкций и элементов исходного кода.

`Моноширинный полужирный шрифт` – используется для обозначения команд или фрагментов текста, которые пользователь должен ввести дословно без изменений, а также в листингах программ, если необходимо обратить особое внимание на фрагмент кода.

`Моноширинный курсив` – применяется для обозначения в исходном коде или в командах шаблонных меток-заполнителей, которые должны быть заменены соответствующими контексту реальными значениями.

Отзывы и пожелания

Мы всегда рады отзывам наших читателей. Расскажите нам, что вы думаете об этой книге – что понравилось или, может быть, не понравилось. Отзывы важны для нас, чтобы выпускать книги, которые будут для вас максимально полезны.

Вы можете написать отзыв прямо на нашем сайте **www.dmkpress.com**, зайдя на страницу книги, и оставить комментарий в разделе «Отзывы и рецензии». Также можно послать письмо главному редактору по адресу **dmkpress@gmail.com**, при этом напишите название книги в теме письма.

Если есть тема, в которой вы квалифицированы, и вы заинтересованы в написании новой книги, заполните форму на нашем сайте по адресу **http://dmkpress.com/authors/publish_book/** или напишите в издательство по адресу **dmkpress@gmail.com**.

Скачивание исходного кода примеров

Скачать файлы с дополнительной информацией для книг издательства «ДМК Пресс» можно на сайте **www.dmkpress.com** на странице с описанием соответствующей книги.

Список опечаток

Хотя мы приняли все возможные меры для того, чтобы удостовериться в качестве наших текстов, ошибки все равно случаются. Если вы найдете ошибку в одной из наших книг – возможно, ошибку в тексте или в коде, – мы будем очень благодарны, если вы сообщите нам о ней. Сделав это, вы избавите других читателей от расстройств и поможете нам улучшить последующие версии данной книги.

Если вы найдете какие-либо ошибки в коде, пожалуйста, сообщите о них главному редактору по адресу **dmkpress@gmail.com**, и мы исправим это в следующих тиражах.

Нарушение авторских прав

Пиратство в интернете по-прежнему остается насущной проблемой. Издательства «ДМК Пресс» и Packt очень серьезно относятся к вопросам защиты авторских прав и лицензирования. Если вы столкнетесь в интернете с незаконно выполненной копией любой нашей книги, пожалуйста, сообщите нам адрес копии или вебсайта, чтобы мы могли применить санкции.

Пожалуйста, свяжитесь с нами по адресу электронной почты **dmkpress@gmail.com** со ссылкой на подозрительные материалы.

Мы высоко ценим любую помощь по защите наших авторов, помогающую нам предоставлять вам качественные материалы.

Глава 1

Введение в обработку естественного языка

Обработка естественного языка (natural language processing, NLP) является важным инструментом для понимания и обработки огромного объема неструктурированных данных в современном мире. В последнее время глубокое обучение широко применяется в NLP, потому что алгоритмы глубокого обучения чрезвычайно эффективно решают задачи классификации изображений, распознавания речи и генерации реалистичных текстов. TensorFlow, в свою очередь, является одной из наиболее простых для освоения и эффективных систем глубокого обучения, существующих в настоящее время. Эта книга адресована начинающим разработчикам систем глубокого обучения и рассказывает про обработку больших объемов данных с использованием NLP и TensorFlow.

В этой главе мы познакомимся с понятием NLP и ответим на вопрос «Что такое обработка естественного языка?». Также мы рассмотрим некоторые из наиболее важных применений NLP. Мы исследуем как традиционные подходы к NLP, так и более поздние подходы на основе глубокого обучения, включая *полносвязные нейронные сети* (fully-connected neural network, FCNN). Глава завершается обзором остальной части книги и программных инструментов, которые мы будем использовать.

Что такое обработка естественного языка?

По данным IBM, в 2017 г. каждый день вырабатывалось 2,5 эксабайта (1 эксабайт = 1 млрд гигабайт) данных, и это число постоянно растет, пока вы читаете данную книгу. Если бы все люди в мире взялись за обработку такого объема данных, то на каждого жителя планеты приходилось бы по 300 МБ в день. В основном эти данные состоят из неструктурированного текста и речи, ведь люди каждый день создают миллионы электронных писем и записей в социальных сетях, а также разговаривают по телефону.

Эта статистика красноречиво говорит нам о том, что такое NLP. Проще говоря, цель NLP – научить компьютер понимать нашу разговорную и письменную речь. Более того, мы уже регулярно используем NLP в повседневной жизни. *Виртуальные помощники* (virtual assistant, VA), такие как Google Assistant, Cortana и Apple Siri, в основном представляют собой системы NLP. Когда кто-то просит виртуаль-

ного помощника: «Покажи мне хороший итальянский ресторан поблизости», – за кулисами сервиса решается множество сложных задач. Во-первых, помощник должен преобразовать речь в текст. Далее он должен понять *семантику запроса* (пользователь ищет хороший ресторан с итальянской кухней) и сформулировать *структурированный запрос* (например, кухня = итальянская, рейтинг = 3–5, расстояние < 10 км). Затем помощник должен отфильтровать известные рестораны по местоположению и кухне. И наконец, нужно отсортировать оставшиеся рестораны по общему рейтингу. Чтобы рассчитать общий рейтинг ресторана, хорошая система NLP может посмотреть как оценки, так и текстовые отзывы, оставленные предыдущими посетителями. Наконец, когда пользователь оказался в ресторане, помощник может перевести различные пункты меню с итальянского на язык пользователя. Этот пример показывает, что обработка естественного языка стала неотъемлемой частью человеческой жизни.

Следует понимать, что NLP – это невероятно сложная область исследований, поскольку слова и смысловое наполнение имеют неоднозначную и нелинейную взаимосвязь, и подобная информация очень плохо поддается представлению в цифровом виде. Что еще хуже, каждый язык имеет свою собственную грамматику, синтаксис и словарный запас. Поэтому обработка текстовых данных включает в себя различные сложные задачи, такие как синтаксический анализ текста (например, токенизация и выделение основы), морфологический анализ, устранение неоднозначности смысла слов и понимание грамматической структуры языка. Например, в предложениях «*Я ничего не должен банку*» и «*Я разбил стеклянную банку*» слово «*банку*» имеет два совершенно разных значения. Чтобы распознать актуальный смысл слова, нам необходимо понять контекст, в котором оно используется. Машинное обучение стало ключевым инструментом NLP, помогая решать упомянутые задачи с помощью машин.

Задачи обработки естественного языка

NLP имеет множество прикладных применений. Хорошая система NLP – это система, которая решает комплекс задач. Когда вы просите Google озвучить прогноз погоды или используете Google Translate, чтобы узнать написание фразы на французском языке, вы запускаете решение цепочки задач NLP. Мы перечислим некоторые из наиболее распространенных задач и расскажем в этой книге об их решении.

- **Токенизация** (tokenization) – это задача разделения *текстового корпуса* (text corpora, большой набор текстовых документов) на неделимые единицы, например слова. Несмотря на обманчивую простоту, токенизация – это важная задача. Например, в японском языке слова не разделяются ни пробелами, ни знаками препинания.
- **Устранение неоднозначности слов** (word-sense disambiguation, WSD) – это задача определения правильного значения слова. Например, в предложениях «*Кредитная карта заблокирована*» и «*Политическая карта Африки*» слово «*карта*» имеет два разных значения. WSD имеет решающее значение для таких задач, как ответы на вопросы.
- **Выделение именованных сущностей** (named entity recognition, NER) – пытается извлечь сущности (например, человека, местоположение и органи-

зацию) из заданного текста или текстового корпуса. Например, предложение «*Джон дал Мэри два яблока в школе в понедельник*» будет преобразовано в [*Джон*]$_{имя}$ дал [*Мэри*]$_{имя}$ [*два*]$_{число}$ яблока в [*школе*]$_{организация}$ в [*понедельник*]$_{время}$. Без NER невозможно обойтись в таких областях, как поиск информации и представление знаний.

- **Морфологическая разметка** (part of speech tagging, PoS) – это задача определения частей речи в предложении и их аннотирование. Это могут быть *основные теги*, например существительное, глагол, прилагательное, наречие и предлог, или же *гранулированные теги*, такие как собственное существительное, имя нарицательное, фразовый глагол и т. д.
- **Классификация предложений/синопсисов** (sentence/synopsis classification). Классификация *предложений* или *синопсисов* (например, обзоров фильмов) имеет множество вариантов использования, таких как обнаружение спама, классификация новостных статей (например, политические, технологические и спортивные) и распознавание отзывов о продукте (например, положительные или отрицательные). Это достигается обучением *модели классификации* на помеченных данных (то есть на обзорах, аннотированных людьми).
- **Генерация естественного языка**. Компьютерная модель, например нейронная сеть, с помощью текстового корпуса обучается генерации новых текстов. Например, можно сгенерировать совершенно новый научно-фантастический рассказ, используя для обучения модели существующие рассказы.
- **Вопросно-ответные системы** (question answering, QA). Технологии вопросно-ответных систем имеют высокую коммерческую ценность и лежат в основе чат-ботов и виртуальных помощников (например, Google Assistant и Apple Siri). Чат-боты широко используются для ответов на вопросы и решения простых проблем клиентов (например, изменения тарифного плана мобильной связи), которые могут быть выполнены без вмешательства человека. Реализация QA-систем охватывает обширные аспекты NLP, такие как поиск информации и представление знаний. Разработка полноценных QA-систем – это сложный и дорогостоящий процесс.
- **Машинный перевод** (machine translation, MT) – это задача преобразования предложения/фразы из исходного языка (например, немецкого) в целевой язык (например, английский). Это очень сложная задача, поскольку разные языки имеют очень разные морфологические структуры, следовательно, это не взаимно однозначное преобразование. Кроме того, межсловные отношения между языками могут строиться по схеме один-ко-многим, один-к-одному, многие-к-одному или многие-ко-многим. В публикациях про MT это принято называть *задачей выравнивания слов* (word alignment problem).

Наконец, в прикладной системе, помогающей человеку в повседневных делах (например, речевой помощник или чат-бот), многие из этих задач должны выполняться совместно. Как мы видели в предыдущем примере, если пользователь просит: «*Покажи мне хороший итальянский ресторан поблизости*», необходимо выполнить несколько различных задач NLP, таких как преобразование устной речи в текст, семантический анализ и анализ настроений, ответы на вопросы и машин-

ный перевод. На рис. 1.1 представлена таксономия задач NLP. Набор задач делится на две широкие категории: *анализ* существующего текста и *генерация* нового текста. В свою очередь, анализ разделяется на три подкатегории: *синтаксический* (задачи, основанные на структуре языка), *семантический* (задачи, основанные на значении) и *прагматический* (открытые проблемы, которые трудно решить):

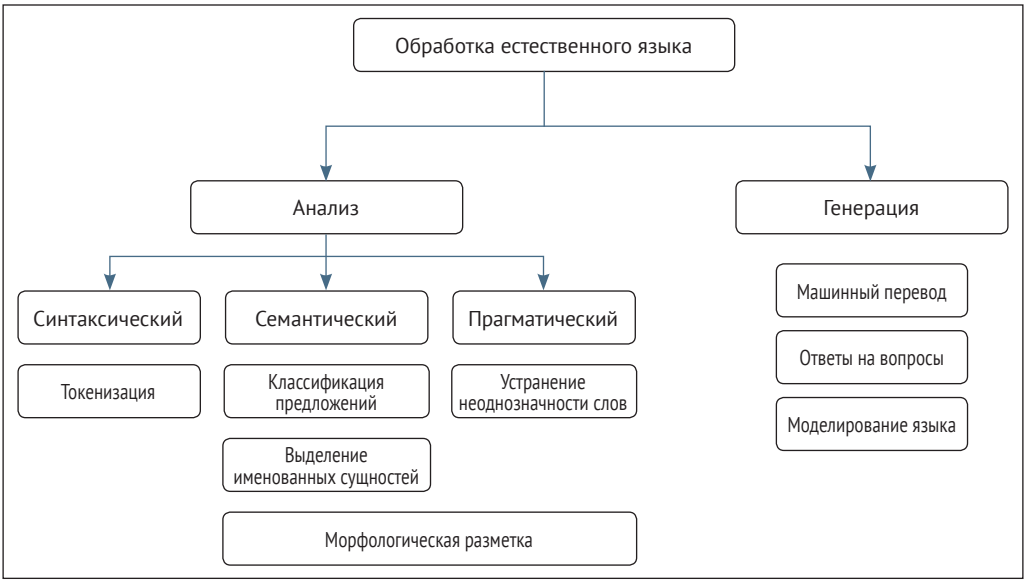

Рис. 1.1 ❖ Таксономия популярных задач NLP на основе наиболее обширных категорий

Разобравшись с типами задач в NLP, давайте теперь перейдем к обсуждению того, как мы можем решать эти задачи с помощью машин.

Традиционный подход к обработке естественного языка

Традиционный подход к NLP основан на статистике и представляет собой последовательность из нескольких ключевых этапов. Фактически это самостоятельные задачи – предварительная обработка, конструирование признаков, обучение модели с помощью обучающих данных и прогнозирование с незнакомыми данными. Наиболее трудоемким и решающим шагом, от которого зависит эффективность и качество NLP, является конструирование признаков.

Подробности традиционного подхода

Традиционный подход к решению задач NLP включает в себя набор отдельных подзадач. Во-первых, текстовые корпусы должны быть предварительно обработаны с целью сокращения *словарного запаса* (vocabulary) и удаления *помех* (distractions). Под помехами я подразумеваю вещи (например, знаки пунктуации

и *стоп-слова*[1]), которые отвлекают алгоритм от сбора важной лингвистической информации, необходимой для выполнения задачи.

Далее следует несколько этапов *конструирования признаков* (feature engineering). Основная задача конструирования признаков – облегчить обучение алгоритмов. Часто эти признаки создаются вручную и смещены в сторону человеческого понимания языка. Конструирование признаков имеет огромное значение для классических алгоритмов NLP, и, следовательно, наиболее эффективные системы обычно имеют более проработанные наборы признаков. Например, для задачи классификации настроений можно представить предложение с помощью *дерева синтаксического разбора* (parse tree) и назначить положительные, отрицательные или нейтральные метки каждому узлу/поддереву в дереве, чтобы классифицировать это предложение как положительное или отрицательное. Кроме того, для конструирования более совершенных признаков можно использовать внешние ресурсы, такие как WordNet (лексическая база данных). Вскоре мы рассмотрим простую технику конструирования признаков, известную как *мультимножество слов* (bag-of-words, BOW).

Затем алгоритм учится хорошо выполнять поставленную задачу, используя полученные признаки и, при необходимости, внешние ресурсы. Например, для задачи *обобщения текста* (text summarization) полезным внешним ресурсом может служить тезаурус, содержащий синонимы слов. И наконец, выполняется прогнозирование: вы берете входные данные, подаете их на вход обученной модели и получаете выходные данные – прогноз. Традиционный подход в общем виде изображен на рис. 1.2.

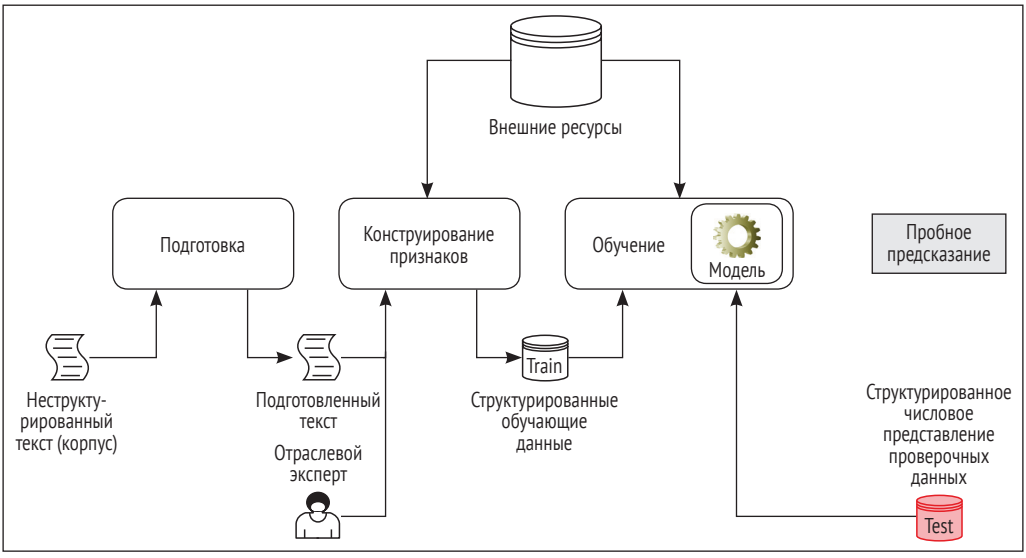

Рис. 1.2 ❖ Традиционный подход к задаче NLP

[1] Стоп-слова – это лексические единицы текста, лишенные смысловой нагрузки (вводные слова, предлоги, междометия и т. д.). – *Прим. перев.*

Пример: создание обзора футбольной игры

Чтобы глубже понять традиционный подход к NLP, рассмотрим задачу автоматической генерации текста на основе статистики футбольного матча. У нас есть несколько наборов показателей игры (например, счет, пенальти и желтые карточки) и соответствующие статьи, написанные для этой игры журналистом, в качестве обучающих данных. Давайте также предположим, что для данной игры у нас есть сопоставление каждого показателя с наиболее релевантной фразой статьи. Наша задача заключается в том, чтобы на основе показателей новой игры сгенерировать обзорную статью об этой игре на естественном языке. Конечно, при наличии достаточно обширного обучающего набора задачу можно решить в лоб: найти для новой игры наиболее похожие показатели в обучающих данных и взять оттуда соответствующую готовую статью. Но есть более сложные и элегантные способы генерации текста.

Если для генерации статьи на естественном языке мы используем машинное обучение, то наверняка будем последовательно выполнять предварительную обработку текста, токенизацию, конструирование признаков, обучение и прогнозирование.

Предварительная обработка текста включает в себя такие операции, как *стемминг*[1] (stemming) и удаление знаков препинания, чтобы уменьшить словарный запас (то есть количество признаков), тем самым уменьшив требования к памяти. Может показаться, что стемминг – это примитивная операция, основанная на простом наборе правил, таких как удаление приставок, суффиксов и окончаний (например, стемминг слова «*прослушивание*» дает стемму «*слуш*»); тем не менее для хорошего алгоритма стемминга требуется нечто большее, чем простая база правил, поскольку иногда правило не очевидно. Допустим, попробуйте из «*наличия весомой аргументации*» алгоритмически получить стемму «*аргумент*». Кроме того, усилия, необходимые для правильного нахождения стеммы, могут различаться по сложности в зависимости от языка.

Токенизация – это процесс разделения корпуса на мелкие объекты, например слова. Токенизация выглядит тривиальной для такого языка, как английский, поскольку слова изолированы; однако это не относится к таким языкам, как тайский, японский и китайский, поскольку в них нет разделения на слова.

Конструирование признаков применяется для преобразования необработанных текстовых данных в оговоренный числовой формат, чтобы можно было обучить модель на этих данных, например преобразовать текст в мультимножество слов или использовать представление в виде *n*-граммы, которое мы обсудим позже. Но учтите, что современные классические модели основаны на гораздо более сложных приемах конструирования признаков. Давайте рассмотрим несколько примеров.

Мультимножество слов – это методика конструирования признаков, которая создает представления элементов на основе частоты появления слов. Допустим, у нас есть два предложения:

- *Боб пошел на рынок, чтобы купить цветы.*
- *Боб купил цветы, чтобы подарить Мэри.*

---

[1] Стемминг – это нахождение основы слова (стеммы), передающей его лексическое значение. – *Прим. перев.*

На основе этих предложений мы можем составить следующий словарь:

[«Боб», «пошел», «на», «рынок», «чтобы», «купить», «цветы», «купил», «подарить», «Мэри»]

Далее мы создадим вектор признаков с размерностью *V* (размер словаря) для каждого предложения, показывающий, сколько раз каждое словарное слово появляется в предложении. В этом примере векторы признаков для предложений будут следующими:

[1, 1, 1, 1, 1, 1, 1, 0, 0, 0]
[1, 0, 0, 0, 1, 0, 0, 1, 1, 1]

К сожалению, в данном методе теряется контекстная информация о порядке слов.

n-грамма – это еще одна техника конструирования признаков, которая разбивает текст на более мелкие компоненты, состоящие из *n* букв или слов. Например, 2-грамма разбивает текст на компоненты, состоящие из двух букв или двух слов. Давайте разложим на 2-граммы знакомое предложение

Боб пошел на рынок, чтобы купить цветы.

Разложение на 2-граммы уровня букв для этого предложения выглядит следующим образом:

[«Бо», «об», «б «, « п», «по», «ош», …, «ит», «ть», «ь «, « ц», «цв», «ве», «ет», «ты»]

Основанное на словах разложение на 2-граммы выглядит так:

[«Боб пошел», «пошел на», «на рынок», «рынок чтобы», «чтобы купить», «купить цветы»]

Преимущество *n*-грамм низкого уровня заключается в том, что словарный запас для больших корпусов будет значительно меньше, чем при использовании слов в качестве признаков.

Затем нам нужно структурировать наши данные, чтобы иметь возможность использовать их для обучения модели. Например, у нас будут кортежи данных вида (`статистика`, `поясняющая фраза`):

(Общее количество голов = 4, Каждая команда забила по два гола в конце первого тайма)

(Команда 1 = Манчестер Юнайтед, Игра между командами Манчестер Юнайтед и Барселона)

(Голы команды 1 = 5, Манчестер Юнайтед сумели забить 5 голов)

Процесс обучения может состоять из трех блоков: *скрытая марковская модель* (hidden Markov model, HMM), *планировщик фразы* (sentence planner) и *планировщик дискурса* (discourse planner). В нашем примере HMM изучает морфологическую структуру и грамматические свойства языка, анализируя совокупность связанных

фраз. Сначала мы изучаем каждую фразу в нашем наборе данных и формируем пары, где первым элементом идет статистика, а за ней следует поясняющая фраза. Затем мы обучаем HMM, попросив его предсказать следующее слово с учетом текущей последовательности. Сначала мы вводим статистику в HMM, а затем получаем прогноз. Далее объединяем последний прогноз с текущей последовательностью и просим HMM дать следующий прогноз и т. д. Таким образом, HMM выводит длинные значимые фразы с учетом статистики.

Далее, у нас может следовать планировщик фраз, исправляющий любые лингвистические ошибки, которые могут встретиться во фразах. Например, планировщик заменяет фразу «*я идти в дом*» фразой «*Я иду домой*». Он может использовать базу данных правил, описывающих правильные способы передачи значений (например, отсутствие предлога между глаголом «*иду*» и существительным «*домой*»).

Теперь мы можем сгенерировать набор фраз для данного набора статистики, используя HMM. Затем нам нужно объединить эти фразы таким образом, чтобы обзор, составленный из набора фраз, был удобочитаемым и правильно передавал ход событий. Допустим, на выходе планировщика фраз мы получили три предложения:

1. Игрок номер 10 команды «Барселона» забил гол во втором тайме.
2. «Барселона» сыграла с «Манчестер Юнайтед».
3. Игрок номер 3 из «Манчестер Юнайтед» получил желтую карточку в первом тайме.

Это лингвистически правильные предложения, но их порядок явно не соответствует логике текста. Намного лучше, когда предложения расположены в таком порядке:

«Барселона» сыграла с «Манчестер Юнайтед». В первом тайме игрок номер 3 из «Манчестер Юнайтед» получил желтую карточку, а во втором тайме игрок номер 10 из «Барселоны» забил гол.

Для упорядочивания и структурирования набора предложений мы используем планировщик дискурса.

Теперь мы можем взять набор произвольных статистик и получить текст обзора футбольного матча, пошагово выполняя рабочий процесс (рис. 1.3).

Имейте в виду, что это объяснение очень высокого уровня, охватывающее только основные компоненты общего назначения, которые, скорее всего, будут включены в традиционный подход к решению задачи NLP. Детали могут существенно различаться в зависимости от конкретной задачи. Для некоторых задач могут потребоваться дополнительные критически важные компоненты, например база правил и модель выравнивания в машинном переводе. Мы не будем углубляться в такие подробности, потому что цель этой книги – рассказать о более современных способах обработки естественного языка.

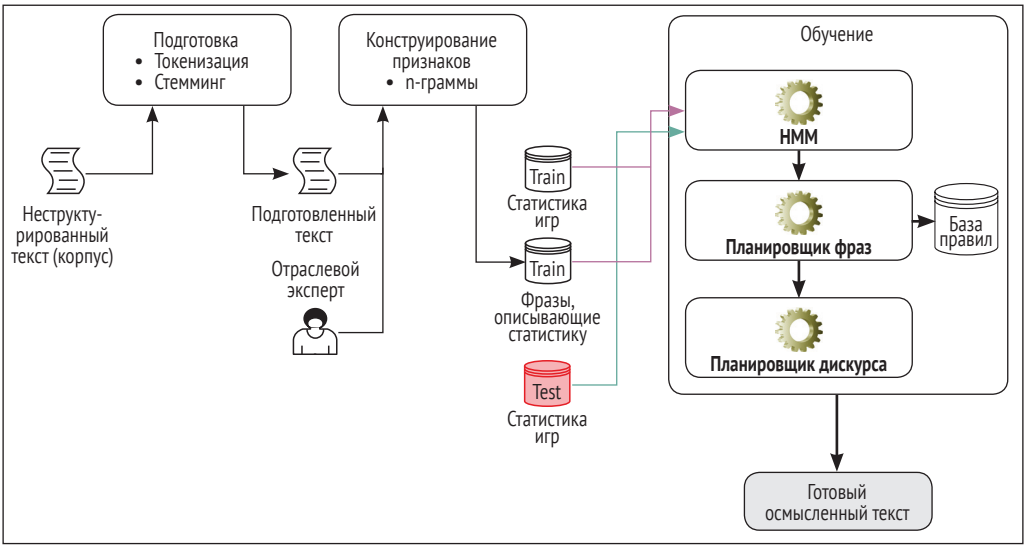

Рис. 1.3 ❖ Этап из примера классического подхода к решению задачи моделирования языка

Недостатки традиционного подхода

Давайте перечислим несколько ключевых недостатков традиционного подхода, поскольку они послужат хорошей основой для понимания причин перехода к глубокому обучению:

- необходимость предварительной обработки текста в традиционном подходе NLP вынуждает искать компромисс между размером словарного запаса и потенциально полезной информацией, встроенной в текст (например, пунктуацией и эмоциональной окраской). Хотя предварительная обработка все еще используется в современных решениях, основанных на глубоком обучении, она не столь важна благодаря большой репрезентативной способности глубоких сетей;
- конструирование признаков должно выполняться вручную. Чтобы получить качественную систему, необходимо сконструировать хорошие признаки. Этот процесс может быть очень трудоемким, так как необходимо тщательно исследовать различные пространства признаков. Кроме того, для эффективного изучения надежных признаков требуется экспертиза предметной области, которая доступна не для всех задач NLP;
- для нормальной работы метода требуются различные внешние ресурсы, но свободно доступных ресурсов не так уж много. Такие внешние ресурсы зачастую состоят из подготовленной вручную информации, хранящейся в больших базах данных. Создание базы данных (например, базы правил машинного перевода) может занять несколько лет, в зависимости от сложности задачи.

Революция глубокого обучения в обработке естественного языка

Думаю, никто не станет спорить, что глубокое обучение произвело революцию в машинном обучении, особенно в таких областях, как компьютерное зрение, распознавание речи и, конечно, NLP. Глубокое обучение вызвало волну смены парадигм во многих областях машинного обучения, потому что глубокие модели умеют самостоятельно извлекать мощные признаки из необработанных данных вместо использования ограниченных искусственных признаков. Это привело к тому, что утомительное и затратное ручное конструирование признаков устарело. Глубокие модели повысили эффективность рабочего процесса, поскольку они одновременно извлекают признаки и учатся решать задачу. Более того, благодаря огромному количеству параметров (т. е. весов) в глубокой модели она может учитывать значительно больше признаков, чем мог бы придумать человек. Однако глубокие модели считаются «черным ящиком» из-за плохой интерпретируемости результатов. Например, понимание того, *как* и *что* использует глубокая модель для решения определенной задачи, все еще остается открытой проблемой.

Глубокая модель – это, по сути, искусственная нейронная сеть, имеющая входной слой, множество взаимосвязанных скрытых слоев в середине и, наконец, выходной слой (например, классификатор или регрессор). Как видите, получается *сквозная модель* (end-to-end model), охватывающая весь процесс, от необработанных данных до прогнозов. Скрытые слои в середине – это сердце глубокой модели, поскольку они отвечают за извлечение полезных признаков из необработанных данных, что в конечном итоге приводит к выполнению поставленной задачи.

История глубокого обучения

Давайте кратко обсудим происхождение глубокого обучения и то, как эта область превратилась в очень многообещающую технологию. В 1960 году Хьюбел (Hubel) и Визель (Wiesel) провели интересный эксперимент и обнаружили, что зрительная кора кошки состоит из простых и сложных клеток и что эти клетки организованы в иерархической форме. Также эти клетки по-разному реагируют на разные раздражители. Например, простые клетки срабатывают в ответ на простые визуальные сигналы, такие как ориентация границ, в то время как сложные клетки не чувствительны к пространственной ориентации объекта. Это открытие разожгло интерес ученых к имитации подобного поведения в машинах, породив концепцию глубокого обучения.

В последующие годы нейронные сети привлекли внимание многих исследователей. В 1965 году Ивахненко и его коллеги представили нейронную сеть, обученную по *методу группового учета аргументов* (group method of data handling, GMDH), основанному на знаменитом *персептроне Розенблатта*. Позже, в 1979 году, Фукусима (Fukushima) представил *неокогнитрон* (neocognitron), который лег в основу одного из самых известных вариантов глубоких моделей – сверточной нейронной сети. В отличие от персептронов, которые всегда обладают одномерным входом, неокогнитрон мог обрабатывать двумерные входы, используя операции свертки.

Искусственные нейронные сети используют алгоритм *обратного распространения ошибки* для оптимизации весовых коэффициентов сети путем вычисления якобиана предшествующего слоя. Кроме того, проблема *исчезающих градиентов* (vanishing gradient) жестко ограничивает потенциальную *глубину*, т. е. количество слоев нейронной сети. Чем ближе слой расположен ко входу, тем меньше становится его градиент, т. е. изменение весов в процессе распространения ошибки, что известно как явление исчезающего градиента. Это происходит по причине цепного вычисления градиентов весов нижних слоев путем последовательного перемножения малых величин.

Затем в 2006 году было обнаружено, что предварительное обучение глубокой нейронной сети на сжатых и нормализованных данных для каждого слоя сети по отдельности обеспечивает хорошие начальные значения весовых коэффициентов, особенно для начальных слоев, что позволяет, по сути, устранить эффект исчезающего градиента. Кроме того, эти более глубокие модели смогли превзойти традиционные модели машинного обучения во многих задачах, в основном в компьютерном зрении. Например, точность распознавания стандартного набора рукописных цифр MNIST близка к 100 %. После этого прорыва глубокое обучение стало модным словом в сообществе машинного обучения.

Глубокие нейросети начали резко набирать обороты, когда в 2012 году Алекс Крижевский (**http://www.cs.toronto.edu/~kriz/**), Илья Суцкевер (**http://www.cs.toronto.edu/~ilya/**) и Джефф Хинтон (Geoff Hinton) представили глубокую сверточную нейросеть AlexNet и выиграли соревнование по визуальному распознаванию изображений 2012 года, уменьшив ошибку на 10 % по сравнению с предыдущим победителем. Примерно в это же время глубокие нейронные сети достигли современного уровня точности распознавания речи. Кроме того, разработчики нейросетей обнаружили, что *графические процессоры* (graphical processor unit, GPU) за счет своей архитектуры обеспечивают отличную параллельность вычислений по сравнению с *центральными процессорами* (central, processor unit, CPU), что позволяет быстрее обучать большие и более глубокие сети.

Глубокие модели были дополнительно улучшены с помощью более совершенных методов инициализации, например *инициализации Завьера* (Xavier), что делает ненужным длительное предварительное обучение. Кроме того, были предложены более подходящие нелинейные функции активации, такие как *выпрямленная линейная функция* (rectified linear unit, ReLU), которые ослабили вредный эффект исчезающего градиента в более глубоких моделях. Более эффективные методы обучения, такие как Adam, автоматически подстраивают индивидуальные скорости обучения каждого параметра среди миллионов параметров, которые мы имеем в модели нейронной сети, что радикально увеличивает производительность нейросетей в таких ресурсоемких областях, как классификация объектов и распознавание речи. Благодаря этим улучшениям удалось увеличить количество скрытых слоев, т. е. сделать нейросети более глубокими. Глубина нейросети является одним из основных факторов, определяющих значительно более высокую точность глубоких моделей по сравнению с другими моделями машинного обучения. Кроме того, улучшенные промежуточные нормализаторы, такие как *слои пакетной нормализации* (batch normalization layer), увеличили производительность глубоких сетей для многих задач.

Позже были представлены феноменально глубокие модели, такие как ResNet, Highway Net и Ladder Net, которые имели сотни слоев и миллиарды параметров.

Это стало возможным благодаря оригинальному сочетанию результатов теоретических и экспериментальных исследований. Например, ResNet использует механизм ссылок для соединения слоев, расположенных далеко друг от друга, что сводит к минимуму послойное уменьшение градиентов, о котором говорилось ранее.

Современное состояние глубокого обучения и NLP

Много разных глубоких моделей увидело свет с момента их появления в начале 2000 года. Несмотря на то что все они используют нелинейное преобразование входных данных и параметров, детали реализации могут сильно различаться. Например, *сверточные нейронные сети* (convolution neural network, CNN) могут извлекать признаки непосредственно из двумерных данных (например, изображений RGB), в то время как многослойный персептрон требует, чтобы входные данные были развернуты в одномерный вектор, что приводит к потере важной пространственной информации.

Понимание текста заключается в том, чтобы уметь интерпретировать его как последовательность символов. Поэтому глубокая модель должна уметь выполнять моделирование временных рядов, что требует памяти прошлых состояний. Чтобы понять это, подумайте о задаче моделирования языка: слову «*кошка*» и слову «*самолет*» обычно предшествуют разные слова. Одна из наиболее популярных моделей, обладающих способностью памяти, известна как *рекуррентная нейронная сеть* (recurrent neural network, RNN). В главе 6 вы увидите, каким образом RNN добиваются этого, выполняя операции с обратной связью.

Следует подчеркнуть, что память – это весьма нетривиальная опция глубокой модели. Способы реализации памяти должны быть тщательно продуманы. Кроме того, термин «память» не следует путать с сохранением весов *одношаговой* глубокой сети, которая смотрит только на текущий вход, тогда как *последовательная* глубокая модель (например, RNN) будет смотреть как на сохраненные веса, так и на предыдущий элемент входной последовательности для прогнозирования следующего вывода.

Одним из существенных недостатков RNN является то, что они могут запомнить лишь несколько – около семи – временных шагов, поэтому им не хватает долговременной памяти. Сети с *долгой краткосрочной памятью* (long short-term memory, LSTM) являются расширением сетей RNN, которые инкапсулируют долговременную память. Поэтому в настоящее время сети с LSTM зачастую предпочтительнее стандартных RNN. Мы обсудим подробности в главе 7.

Таким образом, мы можем разделить глубокие сети на две основные категории: одношаговые сети, которые имеют дело только с одним входом в один момент времени для обучения и прогнозирования (например, классификация изображений), и последовательные сети, которые работают с последовательностями входов произвольной длины (например, генерация текста, где одно слово является одним входом). Затем мы можем разделить одношаговые сети, также называемые сетями *прямого распространения*, на глубокие – приблизительно около 20 слоев, и очень глубокие сети – более чем сотни слоев. Последовательные сети подразделяются на модели кратковременной памяти (например, RNN), которые могут запоминать только краткосрочные паттерны, и модели долговременной памяти, которые могут запоминать более длинные паттерны. На рис. 1.4 схематически изображена обсуждаемая таксономия. На данном этапе от вас не ждут полного

понимания всех этих моделей глубокого обучения. Пока это лишь иллюстрация разнообразия моделей.

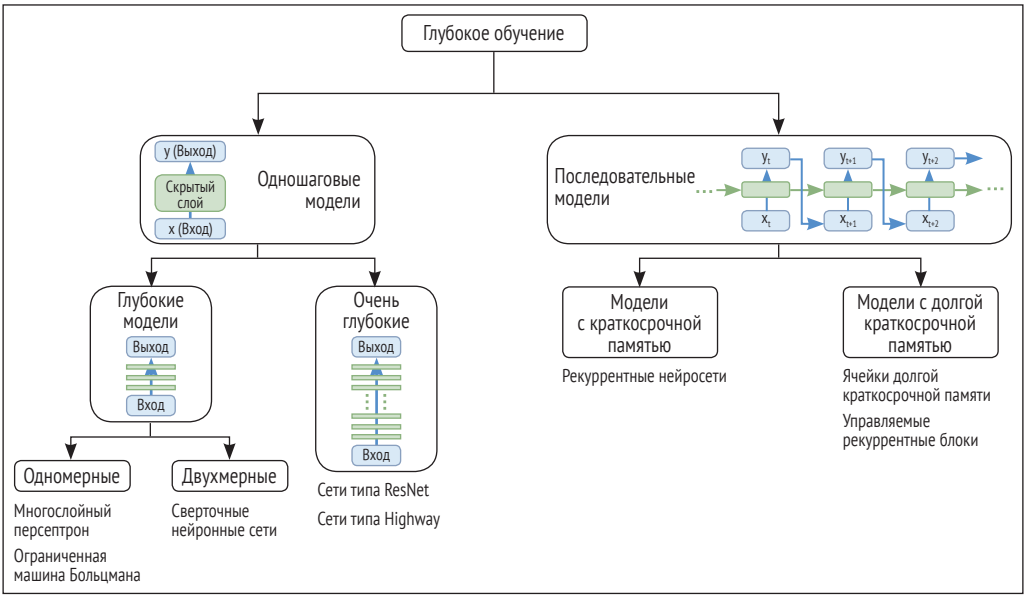

Рис. 1.4 ❖ Общая таксономия наиболее часто используемых методов глубокого обучения

Устройство простой глубокой модели – полносвязной нейронной сети

Теперь давайте подробнее рассмотрим устройство глубокой нейронной сети. Хотя существует множество различных вариантов глубоких сетей, рассмотрим одну из самых ранних моделей 1950–1960 гг., известную как *полносвязная нейронная сеть* (fully connected neural network, FCNN), или *многослойный персептрон*. На рис. 1.5 изображена стандартная трехслойная нейросеть.

Цель FCNN – выполнить классификацию, то есть сопоставить входные данные (например, изображение или предложение) с определенной меткой или аннотацией, например категорией объекта на картинке. Сначала вычисляют h – скрытое представление ввода x с помощью преобразования $h = sigma(W*x + b)$, где W и b – веса и смещение FCNN соответственно, а *sigma* – сигмоидная функция активации. Далее поверх слоев помещается классификатор, изучающий функции скрытых слоев и выполняющий классификацию входных данных. Классификатор, по сути, является частью FCNN и еще одним скрытым слоем с некоторыми весами W_s и смещением b_s. Кроме того, мы можем рассчитать конечный выход FCNN как $output = softmax(W_s*h + b_s)$. Например, классификатор *softmax* обеспечивает нормализованное представление результатов на уровне классификатора. Меткой считается выходной узел с наибольшим значением *softmax*. Получив результат, мы можем определить *ошибку классификации*, которая рассчитывается как разница между прогнозируемой меткой и фактической меткой. Примером такой функ-

ции ошибки является среднеквадратичная ошибка. Вам не нужно беспокоиться, если вы незнакомы с функциями ошибок. Мы обсудим их в следующих главах. Затем параметры нейронной сети W, b, W_s и b_s оптимизируют с использованием стандартного стохастического оптимизатора, например стохастического градиентного спуска, чтобы уменьшить потери классификации всех входных данных. На рис. 1.5 изображен рабочий процесс для трехслойной полносвязной сети. Мы подробно разберем применение такой модели для задач NLP в главе 3.

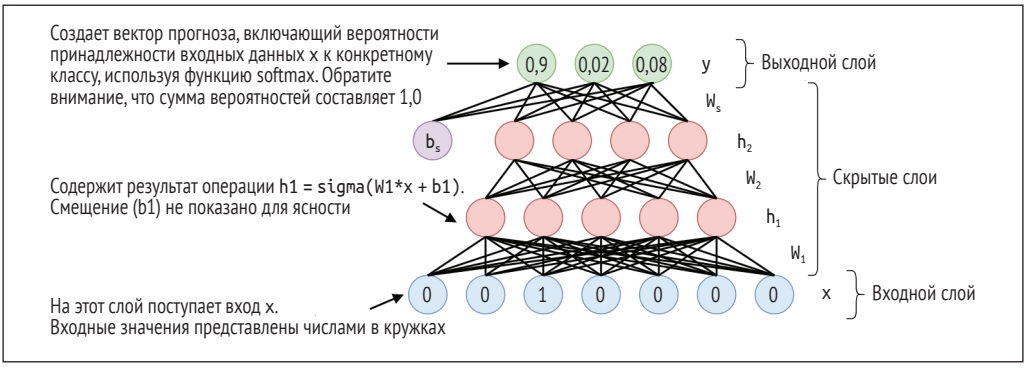

Рис. 1.5 ❖ Пример рабочего процесса полносвязной нейронной сети (FCNN)

Давайте рассмотрим пример использования нейронной сети для задачи *анализа настроений* (sentiment analysis). Предположим, что у нас есть обучающий набор данных в виде предложений на естественном языке, выражающих положительное или отрицательное мнение о фильме. Предложения снабжены метками, означающими, является отзыв положительным (1) или отрицательным (0). Затем нам дают проверочный набор данных, в котором есть отзывы на фильмы в одном предложении, и наша задача – классифицировать эти новые предложения как положительные или отрицательные.

Для этой задачи можно использовать нейронную сеть (которая может быть глубокой или неглубокой в зависимости от сложности задачи), придерживаясь следующего рабочего процесса:

1) токенизируем предложения по словам;
2) при необходимости добавляем к предложениям специальный токен, чтобы привести все предложения к одинаковой длине;
3) переводим предложения в числовое представление, например Bag-of-Words;
4) подаем числовые входы в нейронную сеть и прогнозируем выход (положительный или отрицательный);
5) вычисляем функцию потерь и оптимизируем нейронную сеть.

Что вы найдете дальше в этой книге?

В этом разделе кратко описано содержание остальной части книги. Мы рассмотрим многочисленные интересные области NLP, от алгоритмов, которые находят сходство слов, не используя аннотированные данные, до алгоритмов, которые могут самостоятельно сочинить сказку.

Начиная со следующей главы, мы углубимся в детали нескольких популярных и интересных задач NLP. Чтобы получить глубокие знания и сделать обучение интерактивным, предусмотрены различные упражнения. Мы будем использовать Python и TensorFlow – библиотеку с открытым исходным кодом для распределенных вычислений. TensorFlow воплощает передовые технические решения, такие как оптимизация вашего кода для графических процессоров с использованием технологии Compute Unified Device Architecture (CUDA), что само по себе непросто. Кроме того, TensorFlow предоставляет встроенные функции для реализации алгоритмов глубокого обучения, например активаций, методов стохастической оптимизации и сверток, облегчающие жизнь разработчика.

Итак, мы начинаем путешествие, которое охватывает многие актуальные темы NLP и демонстрирует реализацию самых современных алгоритмов при помощи TensorFlow. Вот что вы найдете дальше в этой книге.

- *Глава 2* дает вам общее понимание того, как писать клиентские программы и запускать их в TensorFlow. Это важно, особенно если вы новичок в TensorFlow, потому что TensorFlow ведет себя иначе, чем традиционный язык программирования, такой как Python. Сначала мы подробно разберем, как в окружении TensorFlow выполняется прикладной код – так называемый *клиент*. Это поможет вам понять рабочий процесс выполнения клиента TensorFlow и чувствовать себя комфортно в новой терминологии. Далее в этой главе вы познакомитесь с различными элементами клиента TensorFlow, такими как определение переменных, определение операций/функций, ввод входных данных в алгоритм и получение результатов. Наконец, мы используем полученные знания на практике и реализуем умеренно сложную нейронную сеть для классификации рукописных изображений.
- *Глава 3* представляет Word2vec – эффективный метод числового представления слов в пространстве смыслов. Но прежде чем углубиться в изучение Word2vec, мы обсудим некоторые классические подходы, используемые для представления семантики слов. Одним из первых подходов было использование WordNet – большой лексической базы данных. WordNet можно применять для измерения семантического сходства между разными словами. Однако поддержание такой большой лексической базы данных является дорогостоящей работой. К счастью, существуют более простые методы представления семантики, не зависящие от внешних ресурсов. Мы рассмотрим эти методы. Затем мы перейдем к современному способу векторного представления слов, известному как Word2vec, где используем нейронную сеть для изучения представлений. Мы обсудим две популярные методики Word2vec: *словосочетания с пропуском* (skip-gram) и *непрерывное мультимножество слов* (continuous bag-of-words, CBOW).
- *Глава 4* начинается с нескольких сравнений, в том числе между алгоритмами skip-gram и CBOW, чтобы узнать, есть ли явный победитель. Затем мы обсудим несколько дополнений, которые были введены в оригинальную технику Word2vec в течение последних нескольких лет. Например, игнорирование распространенных в тексте слов, таких как артикли *the* и *a*, с высокой вероятностью повышает качество моделей Word2vec. С другой стороны, модель Word2vec учитывает только локальный контекст слова и игнорирует глобальную статистику всего корпуса. Отсюда мы приходим к методике

GloVe, которая учитывает как локальную, так и глобальную статистику при поиске векторов слов.

- *Глава 5* знакомит вас со сверточными нейронными сетями (CNN). Сверточные сети – это мощное семейство глубоких моделей, учитывающих пространственную структуру ввода при изучении данных. Другими словами, CNN может обрабатывать непосредственно двумерные изображения, когда многослойному персептрону необходимо развернуть изображение в одномерный вектор. Сначала мы подробно обсудим различные базовые операции CNN, такие как операции свертки и объединения. Затем разберем пример классификации рукописных цифровых изображений с помощью CNN. Далее перейдем к применению сверточных сетей в NLP. Точнее, мы рассмотрим применение CNN для задачи классификации предложений, когда необходимо определить, относится ли предложение к человеку, местоположению, объекту и т. д.
- *Глава 6* посвящена изучению рекуррентных нейронных сетей (RNN) и использованию RNN для генерации естественного языка. Отличительной особенностью рекуррентных сетей является наличие памяти. Память хранится как постоянно обновляемое состояние системы. Мы начнем с представления нейронной сети с прямым распространением и перестроим это представление для изучения последовательных данных. Таким образом, мы превратим сеть прямого распространения в RNN. За этим последует детальное описание уравнений, используемых для вычислений в RNN. Далее мы обсудим процесс обучения RNN и обзорно пройдемся по различным типам RNN, таким как одноранговые и одноконтурные сети. Затем разберем увлекательный пример и научим рекуррентную нейросеть рассказывать новые сказки, извлекая уроки из совокупности существующих сказок. Мы добьемся этого, обучая RNN предсказывать следующее слово с учетом предыдущей последовательности слов сказки. Наконец, обсудим модифицированный вариант нейросети под названием RNN-CF (RNN с контекстными функциями) и сравним его со стандартной RNN, чтобы увидеть, какой вариант работает лучше.
- *В главе 7* мы обсуждаем сети с долгой краткосрочной памятью (LSTM), даем четкое представление о том, как работают эти модели, и постепенно углубляемся в технические детали, достаточные для их самостоятельной реализации. Стандартные рекуррентные сети страдают от невозможности сохранения долговременной памяти. Однако существуют усовершенствованные модели RNN, например ячейки с долгой кратковременной памятью и управляемые рекуррентные блоки (GRU), которые могут запоминать последовательности на большое количество тактов. Мы рассмотрим, как именно LSTM облегчает задачу сохранения долговременной памяти (это называется проблемой исчезающего градиента). Затем обсудим несколько идей для дальнейшего улучшения моделей LSTM, таких как прогнозирование на несколько шагов вперед и чтение последовательностей как вперед, так и назад. Наконец, обсудим несколько вариантов моделей LSTM.
- *Глава 8* рассказывает о практической реализации сетей с долгой краткосрочной памятью. Кроме того, сравним как качественно, так и количественно производительность различных вариантов. Мы также обсудим, как реализовать некоторые расширения, рассмотренные в главе 7, такие как

прогнозирование на несколько шагов вперед (*лучевой поиск,* beam search) и использование векторов слов в качестве входных данных вместо прямого унитарного кодирования. Наконец, обсудим использование API RNN – вспомогательной библиотеки TensorFlow для облегчения реализации рекуррентных моделей.

- *Глава 9* рассказывает про другое интересное приложение, где модель учится генерировать подписи к рисункам, используя LSTM и CNN. Это приложение интересно тем, что показывает нам, как комбинировать модели двух разных типов, а также как распознавать *мультимодальные данные* (например, изображения и текст). Конкретный подход заключается в следующем: сначала получают вектор представления изображения (аналогично векторам слов) с помощью CNN и обучают LSTM, подавая ей на вход вектор изображения, а затем слова описания изображения в виде последовательности. Сначала вы узнаете, как использовать предварительно обученную сверточную сеть для получения представлений изображения. Затем мы обсудим, как получить представление слов. Далее разберемся, как применять векторы изображений вместе с векторами слов для обучения LSTM. Мы перечислим различные метрики оценки систем генерации подписей к изображениям. После этого сможем оценивать подписи, сгенерированные нашей моделью, как качественно, так и количественно. Главу завершает инструкция по внедрению той же системы с помощью RNN API TensorFlow.
- *Глава 10* рассказывает про обучение преобразованию последовательностей и машинный перевод. Машинный перевод привлекает исследователей и разработчиков массовой потребностью людей в автоматическом переводе и сложностью задачи. Мы начнем главу с краткого воспоминания о том, с чего начиналась история машинного перевода, и продолжим введением в системы *нейронного машинного перевода* (neural machine translation, NMT). Мы увидим, насколько хорошо работают современные системы NMT по сравнению со старыми системами статистического машинного перевода. После этого обсудим идею, лежащую в основе устройства систем NMT, и углубимся в технические детали. Затем выберем метрику оценки нашей системы и рассмотрим, как можно реализовать переводчик с немецкого на английский с нуля. Далее вы узнаете о способах улучшения систем NMT. Мы подробно рассмотрим один из подходов, называемый *механизмом внимания* (attention mechanism). Без механизма внимания не обойтись при обучении модели преобразованию последовательности в последовательность. Наконец, мы сравним полученные результаты и проанализируем причины повышения качества при использовании механизма внимания. Главу завершает раздел о том, как можно развить концепцию систем NMT для реализации чат-ботов. Чат-боты – это системы, которые общаются с людьми на естественном языке и помогают выполнять различные запросы пользователей.
- *Глава 11* рассказывает о современных тенденциях и будущем обработки естественного языка. Обработка естественного языка охватывает широкий спектр различных задач. Мы обсудим некоторые текущие тенденции и предположения о том, чего можно ожидать от NLP в будущем. Сначала обсудим различные расширения для обучения смысловой векторизации слов, появившиеся недавно. Мы также рассмотрим реализацию одной из таких

техник обучения векторизации, известную как TV-embedding. Далее разберем различные направления развития в области нейронного машинного перевода. Затем обсудим, как NLP сочетается с другими областями, такими как компьютерное зрение и обучение с подкреплением, для решения некоторых любопытных проблем, таких как обучение компьютерных агентов общению путем разработки собственного языка. В наши дни еще одной важной областью исследований является *общий искусственный интеллект*, связанный с разработкой систем, способных выполнять несколько задач (классифицировать изображения, переводить текст, генерировать подписи к изображениям и т. д.) в рамках одной системы. Мы исследуем несколько таких систем. Затем поговорим о внедрении NLP в майнинг социальных сетей. Мы завершим эту главу примерами некоторых новых задач, например языковым обоснованием – разработкой систем NLP на основе здравого смысла, и новыми моделями, такими как синхронизированные LSTM.

- *Приложение* познакомит читателя с различными математическими структурами данных и операциями (например, с обращением матриц). Мы также обсудим несколько важных понятий теории вероятностей. Затем мы представим Keras – высокоуровневую библиотеку, которая использует TensorFlow. Keras упрощает внедрение нейронных сетей, скрывая некоторые детали реализации TensorFlow, что может показаться достаточно запутанным. Вы познакомитесь с примером реализации сверточной сети с помощью Keras. Далее мы обсудим, как использовать библиотеку seq2seq в TensorFlow для реализации системы машинного перевода с гораздо меньшим количеством кода, чем тот, который вы использовали в главе 11. Наконец, вы познакомитесь с руководством по использованию TensorBoard для визуализации смысловых связей слов. TensorBoard – это удобный инструмент визуализации, который поставляется с TensorFlow. Его можно использовать для визуализации и мониторинга различных переменных в вашем клиенте TensorFlow.

Знакомство с рабочими инструментами

В этом разделе вы познакомитесь с техническими средствами, которые будут задействованы в упражнениях следующих глав. Сначала мы представим обзор основных инструментов. Далее дадим краткие инструкции по установке каждого инструмента вместе с гиперссылками на подробные руководства, предоставленные официальными сайтами. Кроме того, мы поделимся советами о том, как убедиться, что инструменты были установлены правильно.

Обзор основных инструментов

Для написания кода программ мы будем использовать Python. Это очень универсальный, легко настраиваемый язык программирования, который активно используется научным сообществом. Кроме того, вокруг Python существует множество научных библиотек, охватывающих различные области, от глубокого обучения до вероятностного вывода и визуализации данных. TensorFlow – одна из таких биб-

лиотек, хорошо известная в сообществе глубокого обучения и предоставляющая множество базовых и сложных операций, полезных для глубокого обучения.

Во всех наших упражнениях мы будем использовать блокноты Jupyter Notebook, поскольку они обеспечивают более интерактивную среду программирования по сравнению с использованием IDE.

Мы также будем использовать scikit-learn – еще один популярный инструментарий машинного обучения для Python – для различных прикладных целей, таких как предварительная обработка данных. Для различных операций с текстом мы будем использовать NLTK – набор инструментов Python для естественного языка. Наконец, применим пакет Matplotlib для визуализации данных.

Установка Python и scikit-learn

Python без проблем устанавливается в любой из широко используемых операционных систем, таких как Windows, macOS или Linux. Для установки и настройки Python мы будем использовать дистрибутив Anaconda, поскольку он выполняет всю кропотливую работу по настройке Python, а также устанавливает дополнительные пакеты и библиотеки.

Чтобы установить Anaconda, выполните следующие действия:

1) загрузите Anaconda с **https://www.anaconda.com/distribution/**. На момент подготовки русского перевода была доступна версия Anaconda3 2019.07;
2) выберите подходящую ОС и загрузите пакет с поддержкой Python 3.7;
3) установите Anaconda, следуя инструкциям на **https://docs.continuum.io/anaconda/install/**.

Чтобы проверить правильность установки Anaconda, выполните следующие действия:

1) в списке установленных программ найдите приложение **Anaconda Prompt** или **Anaconda Promt Shell** и запустите его;
2) теперь выполните следующую команду:

```
conda --version
```

При правильной установке в окне терминала должна отобразиться версия текущего дистрибутива Anaconda. На момент подготовки перевода это была версия 4.7.10.

Пакеты scikit-learn, NLTK и Matplotlib автоматически устанавливаются в составе нового дистрибутива Anaconda. Затем вы можете обновить версии пакетов. Найдите на своем компьютере приложение Anaconda Navigator и запустите его. Перейдите в окне навигатора на вкладку **Environment** (Окружение). Пакеты, для которых доступно стабильное обновление, помечены стрелкой в столбце **Version** (Версия). Щелкните правой кнопкой мыши на значке галочки и выберите пункт меню **Mark for update** (Отметить для обновления). Отметив нужные пакеты, нажмите кнопку **Apply** (Применить).

Установка Jupyter Notebook

Jupyter Notebook автоматически устанавливается вместе с новым дистрибутивом Anaconda. Вы можете установить Jupyter Notebook вручную, следуя инструкции на странице **http://jupyter.readthedocs.io/en/latest/install.html**.

Чтобы проверить правильность установки Jupyter Notebook, выполните следующие действия:

1) откройте окно терминала Anaconda Prompt;
2) выполните команду

```
jupyter notebook
```

Если новое окно браузера не открывается автоматически, то скопируйте адресную строку с секретным токеном из окна терминала и вставьте ее в адресное поле браузера. В браузере должно открыться рабочее окно, которое выглядит как на рис. 1.6:

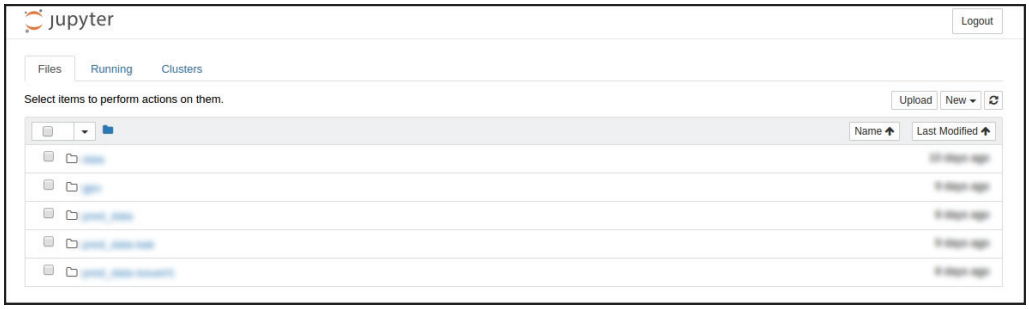

Рис. 1.6 ❖ Juputer Notebook установлен успешно

Установка TensorFlow

Для установки TensorFlow откройте окно терминала Anaconda Prompt и выполните команду

```
pip install tensorflow
```

Дождитесь окончания установки пакетов. На момент подготовки русского перевода книги была доступна версия TensorFlow 1.14.0. Для работы с упражнениями из этой книги нужна версия не ниже 1.8.0, поскольку API претерпел много изменений по сравнению с предыдущими версиями TensorFlow. Если у вас уже установлена версия TensorFlow ниже 1.8.0, обновите ее при помощи команды

```
pip update tensorflow
```

Чтобы проверить правильность установки TensorFlow, выполните следующие действия.

1. В терминале Anaconda Prompt введите команду

   ```
   python
   ```

   для запуска интерпретатора Python. В ответной строке вывода вы должны увидеть версию Python. Убедитесь, что вы используете Python 3.
2. Затем введите следующие команды в строке интерпретатора Python:

   ```
   import tensorflow as tf
   print (tf.__version__)
   ```

Если все прошло хорошо, должна отображаться версия TensorFlow 1.14.0 или выше без сообщений об ошибках. Если вы увидите предупреждение, что на вашем компьютере нет выделенного графического процессора, можете его проигнорировать.

Пользователям доступно множество облачных вычислительных платформ, где вы можете создать свой собственный виртуальный компьютер с различными настройками (операционная система, тип карты GPU, количество карт GPU и т. д.). Многие исследователи переходят на такие облачные сервисы благодаря следующим преимуществам:

- дополнительные параметры настройки;
- меньше усилий по обслуживанию;
- не требуется собственная инфраструктура.

Вот несколько популярных облачных вычислительных платформ:

- облачная платформа Google (GCP): **https://cloud.google.com/**;
- Amazon Web Services (AWS): **https://aws.amazon.com/**;
- TensorFlow Research Cloud (TFRC): **https://www.tensorflow.org/tfrc/**.

Заключение

В этой главе вы получили представление о задачах, связанных с построением хорошей системы на основе NLP. Сначала вы узнали, зачем нужна обработка естественного языка, а затем обсудили различные проблемы и осознали, насколько трудно добиться успеха в решении этих задач.

Далее мы рассмотрели классический подход к реализации NLP на примере генерации текста футбольного репортажа на естественном языке. Мы увидели, что традиционный подход обычно включает кропотливое и утомительное конструирование признаков. Например, чтобы проверить правильность сгенерированной фразы, возможно, придется сгенерировать дерево разбора для этой фразы. Затем мы обсудили смену парадигмы, которая произошла с появлением глубокого обучения, и обнаружили, что глубокое обучение сделало ненужным этап конструирования признаков. Мы начали с небольшого путешествия во времени, чтобы вернуться к истокам глубокого обучения и искусственных нейронных сетей, и постепенно добрались до огромных современных сетей с сотнями скрытых слоев. Позже разобрали простой пример, иллюстрирующий глубокую модель – многослойную модель персептрона, – чтобы прикоснуться к магии математики, скрытой в глубинах нейронных сетей.

Изучив основы традиционных и современных подходов к NLP, мы обсудили дальнейшие темы, которые будут представлены в книге, от представления слова в пространстве смыслов до мощных рекуррентных сетей с памятью, от создания подписей к изображениям до нейронных машинных переводчиков! Наконец, вы настроили на своем компьютере рабочую среду, установив Python, scikit-learn, Jupyter Notebook и TensorFlow.

В следующей главе вы изучите основы TensorFlow. К концу главы вы должны освоить реализацию простого алгоритма, который способен принимать некоторые входные данные, пропускать эти данные через определенную функцию и выводить результат.