Содержание

Предисловие ... 10
От автора ... 11
Благодарности ... 13
Об этой книге ... 14
Об авторе ... 18
Об иллюстрации на обложке ... 19

Часть I. Поиск встречается с глубоким обучением ... 20

Глава 1. Поиск на основе нейронных сетей ... 21
1.1. Нейронные сети и глубокое обучение ... 23
1.2. Что такое машинное обучение? ... 25
1.3. Что глубокое обучение может сделать для поиска ... 27
1.4. Глубокое обучение: дорожная карта ... 30
1.5. Получение полезной информации ... 31
1.5.1. Текст, токены, термы и основы поиска ... 33
1.5.2. Релевантность прежде всего ... 41
1.5.3. Классические модели поиска ... 42
1.5.4. Точность и полнота ... 43
1.6. Нерешенные проблемы ... 43
1.7. Открываем черный ящик поисковой системы ... 45
1.8. Глубокое обучение спешит на помощь ... 46
1.9. Индекс, пожалуйста, познакомьтесь с нейроном ... 50
1.10. Обучение нейронной сети ... 51
1.11. Перспективы поиска на базе нейронных сетей ... 54
Резюме ... 54

Глава 2. Генерируем синонимы ... 56
2.1. Расширение синонимов. Введение ... 57
2.1.1. Почему синонимы? ... 58
2.1.2. Сопоставление синонимов на базе словаря ... 60
2.2. Важность контекста ... 69
2.3. Нейронные сети прямого распространения ... 71
2.4. Использование word2vec ... 75
2.4.1. Настройка word2vec в Deeplearning4j ... 83
2.4.2. Расширение синонимов на базе Word2vec ... 84
2.5. Оценки и сравнения ... 87
2.6. Соображения относительно продукционных систем ... 88
2.6.1. Синонимы против антонимов ... 90

Резюме ... 91

Часть II. Подключение нейронных сетей для использования их в поисковой системе ... 92

Глава 3. От простого поиска к генерации текста ... 93

3.1. Информационная потребность в сравнении с запросом: преодоление разрыва ... 94
3.1.1. Генерация альтернативных запросов ... 95
3.1.2. Подготовка данных ... 97
3.1.3. Подведем итог ... 104
3.2. Обучение на последовательностях ... 105
3.3. Рекуррентные нейронные сети ... 107
3.1.1. Внутреннее устройство и динамика RNC ... 110
3.3.2. Долгосрочные зависимости ... 113
3.3.3. LSTM-сети ... 114
3.4. LSTM-сети для генерации текста без контроля ... 115
3.4.1. Неуправляемое расширение запроса ... 122
3.5. От неконтролируемой до контролируемой генерации текста ... 126
3.5.1. Создание моделей sequence-to-sequence ... 126
3.6. Соображения относительно продукционных систем ... 129
Резюме ... 130

Глава 4. Более чувствительные поисковые подсказки ... 132

4.1. Генерация поисковых подсказок ... 133
4.1.1. Подсказки при составлении запросов ... 133
4.1.2. Подсказчики на базе словаря ... 134
4.2. Lookup API ... 135
4.3. Проанализированные подсказчики ... 138
4.4. Использование языковых моделей ... 145
4.5. Подсказчики на базе контента ... 149
4.6. Нейронные языковые модели ... 150
4.7. Нейронная языковая модель на базе символов для создания подсказок ... 152
4.8. Настройка языковой модели ... 155
4.9. Вносим разнообразие в подсказки, используя векторные представления слов ... 164
Резюме ... 166

Глава 5. Ранжирование результатов поиска с помощью векторных представлений слов ... 167

5.1. Важность ранжирования ... 168
5.2. Модели поиска ... 170
5.2.1. TF-IDF и модель векторного пространства ... 172
5.2.2. Ранжирование документов в Lucene ... 175
5.2.3. Вероятностные модели ... 178
5.3. Поиск информации на базе нейронных сетей ... 180
5.4. От векторов слов к векторам документов ... 180

5.5. Оценки и сравнения 186
5.5.1. Класс Similarity, основанный на усредненных векторных представлениях слов 188
Резюме 191

Глава 6. Векторные представления документов для ранжирования и рекомендаций 192

6.1. От векторных представлений слов к векторным представлениям документов 193
6.2. Использование векторов абзацев в ранжировании 196
6.2.1. ParagraphVectorsSimilarity 198
6.3. Векторные представления документов и сопутствующий контент 199
6.3.1. Поиск, рекомендации и сопутствующий контент 200
6.3.2. Использование частых термов для поиска похожего контента 201
6.3.3. Извлечение аналогичного контента с помощью векторов абзаца 210
6.3.4. Извлечение аналогичного контента с помощью векторов из моделей «кодер–декодер» 212
Резюме 214

Часть III. Шаг за пределы 215

Глава 7. Поиск по языкам 216

7.1. Обслуживание пользователей, говорящих на нескольких языках 216
7.1.1. Перевод документов в сравнении с переводом запросов 218
7.1.2. Поиск по нескольким языкам 220
7.1.3. Запросы на нескольких языках поверх Lucene 221
7.2. Статистический машинный перевод 223
7.2.1. Выравнивание 225
7.2.2. Перевод на основе фраз 226
7.3. Работа с параллельными корпусами 227
7.4. Нейронный машинный перевод 229
7.4.1. Модели кодер–декодер 230
7.4.2. Модель «кодер–декодер» для машинного перевода в DL4J 233
7.5. Векторные представления слов и документов для нескольких языков 240
7.5.1. Монолингвальные векторные представления с использованием линейной проекции 241
Резюме 246

Глава 8. Поиск изображений на основе контента 247

8.1. Содержимое изображения и поиск 248
8.2. Взгляд назад: поиск изображений на базе текста 251
8.3. Понять изображения 253
8.3.1. Представления изображений 255
8.3.2. Извлечение признаков 257
8.4. Глубокое обучение для представления изображений 266
8.4.1. Сверточные нейронные сети 267
8.4.2. Поиск изображений 275

8.4.3. Локально-чувствительное хеширование 280
8.5. Работа с непомеченными изображениями 283
Резюме 288

Глава 9. Взглянем на производительность 289
9.1. Производительность и перспективы глубокого обучения 290
9.1.1. От проектирования модели до производства 291
9.2. Индексы и нейроны работают вместе 306
9.3. Работа с потоками данных 309
Резюме 315
Глядя вперед 315

Предметный указатель 317

Предисловие

Не просто дать количественную оценку того, насколько обыденными стали такие термины, как *нейронные сети* и *глубокое обучение*, и, более конкретно, как эти технологии влияют на нашу жизнь. От автоматизации рутинных заданий до тиражирования трудных решений, помощи в управлении автомобилем (и людьми) до места назначения – мощь нейронных сетей и глубокого обучения в качестве методов, которые произведут революцию в области вычислений, находится лишь в стадии зарождения.

Вот почему эта книга так важна. Нейронные сети, искусственный интеллект (ИИ) и глубокое обучение не только автоматизируют рутинные задания и решения, облегчая их. Они также облегчают и поиск. Прежде состояние дел в области поиска информации использовало сложную линейную алгебру, включая в себя матричное умножение для обозначения сопоставления пользовательских запросов с документами. Сегодня вместо использования алгебраических и линейных моделей применяются, например, нейронные сети для распознавания сходства слов между документами после изучения способов суммирования документов в слова с использованием обособленных сетей. И это только одна область в процессе поиска, где используются ИИ и глубокое обучение.

В своей книге Томмазо Теофили использует практический подход, чтобы показать вам современное состояние использования нейронных сетей, искусственного интеллекта и глубокого обучения в разработке поисковых систем. Книга изобилует примерами и знакомит читателя с архитектурой современных поисковых систем, а также дает вам достаточно информации, чтобы понять, как и где подходит глубокое обучение и как это улучшает поиск. Автор показывает вам, где искусственный интеллект и глубокое обучение могут перегрузить ваш код и возможность поиска, начиная от создания вашей первой сети для поиска похожих слов в расширении запроса и заканчивая изучением векторного представления слов для поискового ранжирования, а также мультиязычного поиска и поиска по изображениям.

Эта книга написана настоящим первопроходцем в области открытого исходного кода. Томмазо – бывший председатель проекта Apache Lucene – механизма индексации поиска де-факто, который поддерживает Elasticsearch и Apache Solr. Он также внес большой вклад в понимание языков и перевод при разработке библиотеки Apache OpenNLP. Совсем недавно его кандидатура была предложена на пост председателя (инкубационного) проекта Apache Joshua для статистического машинного перевода.

Я знаю, что вы многому научитесь из этой книги, и я рекомендую ее, чтобы вы могли найти золотую середину между здравым смыслом, толкованиями теории сложности и реальным кодом, с которым вы можете экспериментировать, используя новейшие технологии глубокого обучения и поиска.

Наслаждайтесь. Я знаю, о чем говорю, потому что именно это я и делал!

Крис Мэттманн,
заместитель директора по технологиям и инновациям,
Лаборатория реактивного движения НАСА

От автора

Обработка естественного языка околдовала меня, как только я узнал о ней, почти 10 лет назад, когда учился в магистратуре. Обещание, что компьютеры могут помочь нам понять (уже даже тогда) огромное количество существующих текстовых документов, было похоже на волшебство. Я до сих пор помню, как было здорово видеть, как мои первые программы для ОЕЯ извлекают пусть даже смутно правильную и полезную информацию из нескольких текстовых документов.

Примерно в то же время на работе меня попросили проконсультировать клиента по поводу новой архитектуры поиска с открытым исходным кодом. Мой коллега, который был экспертом в этой области, был занят другим проектом, поэтому мне дали копию книги *Lucene в действии*[1], которую я изучал на протяжении пары недель. Затем меня отправили на работу консультантом. Спустя пару лет после того, как я проработал над проектом на базе Lucene/Solr, заработала новая поисковая система (и, насколько я знаю, она используется до сих пор). Не могу сказать, сколько раз нужно было настраивать алгоритмы поисковой системы из-за того или иного запроса или того или иного фрагмента проиндексированного текста, но мы заставили ее работать. Я мог видеть запросы пользователей и мог видеть данные, которые должны были быть извлечены, но минимальная разница в написании или пропуске определенного слова могла привести к тому, что очень релевантная информация не была бы отображена в результатах поиска. Поэтому, хотя я очень гордился своей работой, я продолжал задаваться вопросом, как сделать все возможное, чтобы избежать множества ручных вмешательств, которые менеджеры по программному продукту просили меня выполнить, чтобы обеспечить наилучший опыт взаимодействия.

Сразу же после этого я совершенно случайно оказался вовлеченным в машинное обучение благодаря первому онлайн-занятию по машинному обучению от Эндрю Ына (которое было основано на серии Coursera MOOC). Я был настолько очарован концепциями нейронных сетей, показанными на уроке, что решил самостоятельно попробовать реализовать небольшую библиотеку для нейронных сетей на Java, просто ради удовольствия (**http://svn.apache.org/repos/asf/labs/yay/**). Я начал искать другие онлайн-курсы, такие как курс Андрея Карпати по сверточным нейронным сетям для визуального распознавания и курс Ричарда Сохера по глубоким нейронным сетям для обработки естественного языка. С тех пор я продолжаю работать над поисковыми системами, обработкой естественного языка и глубоким обучением, в основном в открытом исходном коде.

Пару лет назад (!) издательство Manning обратилось ко мне с рецензией на книгу об ОЕЯ, и я был достаточно наивен, чтобы написать в нижней части своего обзора, что мне было бы интересно написать книгу о поисковых системах и нейронных сетях. Когда издательство снова обратилось ко мне, проявив интерес, я был немного удивлен и спросил себя, *действительно* ли я хочу написать книгу об этом? Я понял, что да, мне это было интересно.

---

[1] **http://www.manning.com/books/lucene-in-action-second-edition**.

Несмотря на то что глубокое обучение произвело революцию в компьютерном зрении и обработке естественного языка, многое еще предстоит обнаружить, когда речь идет о приложениях, использующихся в поиске. Я уверен, что мы не можем (пока еще?) полагаться на глубокое обучение, чтобы автоматически настраивать поисковые системы от своего имени, но это может помочь сделать работу пользователя поисковой системы более гладкой. Благодаря глубокому обучению мы можем делать в поисковых системах то, чего пока не можем делать с помощью других существующих методов, и можем использовать глубокое обучение, чтобы улучшить техники, которые мы уже используем в поисковых системах. Путь к тому, чтобы сделать поисковые машины более эффективными с помощью глубоких нейронных сетей, только начался. Надеюсь, вам понравится.

Благодарности

Прежде всего я хотел бы поблагодарить свою любимую жену Микелу за помощь и поддержку на протяжении всего этого долгого путешествия: спасибо за любовь, энергию и преданность делу в течение долгих дней, ночей и выходных, в ходе которых я писал эту книгу!

Спасибо Джакомо и Маттиа за то, что они помогли мне выбрать самую классную иллюстрацию для обложки, а также за те игры и смех, которые сопровождали меня, пока я пытался писать.

Я хотел бы поблагодарить своего отца за то, что он гордится мной, и его веру в меня.

Большое спасибо моему другу Федерико за его неустанные усилия по просмотру всех материалов (книга, код, изображения и т. д.) и за приятные обсуждения и обмен идеями. Огромное спасибо моим друзьям и коллегам Антонио, Франческо и Симоне за их поддержку, смех и советы. Также адресую благодарность своим товарищам по проекту Apache OpenNLP (**http://opennlp.apache.org**), Сунилу, Йорну и Кодзи, за то, что предоставили отзывы, советы и идеи, которые помогли придать очертания этой книге.

Я благодарю Криса Мэтманна за написание такого вдохновляющего пролога.

Я также благодарю Фрэнсис Лефковиц, моего редактора по разработке, за ее терпение и руководство на протяжении всего процесса написания книги, включая наши дискуссии о Стефе, К.Д. и Воинах. И я благодарю других людей из издательства Manning, благодаря которым стало возможным появление этой книги, включая издателя Марьян Бэйс и лиц из редакционной и производственной команд, которые работали за кадром. Кроме того, я благодарю технических рецензентов во главе с Иваном Мартиновичем – Абхинава Упадхая, Эля Кринкера, Альберто Симоеса, Альваро Фалькину, Эндрю Уилли, Антонио Магнаги, Криса Моргана, Джулиано Бертоти, Грега Занотти, Йеруна Бенкхуйзена, Крифа Дэвида, Люка Мартина Бира, Майкла Уолла, Михала Пашкевича, Мирко Кемпфа, Паули Сутелайнен, Симона Русо, Срдана Дукича и Урсина Стаусса – и участников форума. Что касается технической стороны, выражаю благодарность Михилу Тримпе, который был техническим редактором книги, и Карстену Стрёбеку, который работал техническим корректором книги.

Наконец, я хотел бы поблагодарить сообщества Apache Lucene и Deeplearning4j за то, что предоставили такие превосходные инструменты, и за дружескую поддержку пользователей.

Об этой книге

Создание поисковых систем с использованием методов глубокого обучения – это практическая книга о том, как использовать (глубокие) нейронные сети для создания эффективных поисковых систем. В ней рассматривается несколько компонентов поисковой системы, дается представление о том, как они работают, и рекомендации о том, как можно использовать нейронные сети в каждом контексте. Основное внимание уделяется практическому объяснению методов поиска и глубокого обучения на базе примеров. Большинство из них сопровождается кодом. В то же время, когда это необходимо, приводятся ссылки на соответствующие исследовательские работы, чтобы побудить вас читать больше и углублять свои знания по конкретным темам. Нейронные сети и темы, связанные с поиском, объясняются на протяжении всей книги.

Прочитав ее, вы получите четкое представление об основных проблемах, связанных с поисковыми системами, о том, как они обычно решаются, и о том, как в этом может помочь глубокое изучение. Вы получите четкое представление о ряде различных методов глубокого обучения и о том, где они вписываются в контекст поиска. Вы также познакомитесь с библиотеками Lucene и Deeplearning4j. Кроме того, вы выработаете практическое отношение к тестированию эффективности нейронных сетей (вместо того чтобы рассматривать их как волшебство) и измерению их затрат и выгод.

Для кого предназначена эта книга

Данная книга предназначена для читателей, владеющих программированием на среднем уровне. Еще лучше, если вы хорошо разбираетесь в программировании на языке Java, интересуясь или активно участвуя в разработке поисковых систем. Вам следует прочитать эту книгу, если вы хотите сделать свою поисковую систему более эффективной, чтобы предоставлять релевантные результаты и, следовательно, сделать ее более полезной для конечных пользователей.

Даже если у вас нет такого опыта, вы будете знакомиться с основными понятиями, касающимися поисковых систем, на протяжении всей книги, когда будет затрагиваться каждый конкретный аспект поиска. Аналогично вам не обязательно иметь познания в области машинного или глубокого обучения. В этой книге будут представлены все необходимые основы машинного и глубокого обучения, а также практические советы, касающиеся применения глубокого обучения в поисковых системах в случаях реальной эксплуатации.

Вы должны быть готовы взять в руки код и расширить существующие библиотеки с открытым исходным кодом для реализации алгоритмов глубокого обучения для решения задач поиска.

Дорожная карта

Эта книга состоит из трех частей:

- первая часть знакомит вас с основными понятиями поиска, машинного и глубокого обучения. В главе 1 рассказывается об обосновании примене-

ния методов глубокого обучения для поиска проблем, затрагивая проблемы в отношении наиболее распространенных подходов к поиску информации. В главе 2 приводится первый пример того, как использовать модель нейронной сети для повышения эффективности поисковой системы путем генерации синонимов из данных;

- вторая часть посвящена общим задачам поисковых систем, которые можно лучше решать с помощью глубоких нейронных сетей. Глава 3 знакомит вас с использованием рекуррентных нейронных сетей для генерации запросов, альтернативных тем, которые вводят пользователи. Глава 4 посвящена задаче предоставления других предложений, в то время как пользователь набирает запрос, с помощью глубоких нейронных сетей. Глава 5 рассказывает о моделях ранжирования, в частности о том, как предоставить более релевантные результаты поиска, используя векторные представления слов. Глава 6 посвящена использованию векторных представлений документов как в функциях ранжирования, так и в контексте метода фильтрации на основе содержания, используемого при построении рекомендательных систем;
- в третьей части рассматриваются более сложные сценарии, такие как машинный перевод с использованием глубокого обучения и поиск изображений. В главе 7 рассказывается о мультиязычных возможностях поисковой системы с помощью подходов на базе нейронных сетей. Глава 8 посвящена поиску коллекции изображений на основе их содержимого, основанной на моделях глубокого обучения. В главе 9 обсуждаются темы, связанные с реальной эксплуатацией, такие как точная настройка моделей глубокого обучения и работа с постоянно поступающими потоками данных.

Сложность рассматриваемых тем и концепций возрастает в ходе прочтения книги. Если вы новичок в области глубокого обучения, поиска или того и другого, я настоятельно рекомендую сначала прочитать главы 1 и 2. В противном случае не стесняйтесь перепрыгивать и выбирать главы, основываясь на своих потребностях и интересах.

О КОДЕ

В этой книге предпочтение отдается фрагментам кода, а не подробным листингам, чтобы читатель мог быстро и легко понять, что и как делает код. Полный исходный код можно найти на странице книги на сайте издательства Manning: **www.manning.com/books/deep-learning-for-search**. Программное обеспечение также будет обновляться на официальной странице книги на GitHub (**https://github.com/dl4s**), включая исходный код на Java в книге (с использованием Apache Lucene и Deeplearning4j: **https://github.com/dl4s/dl4s**) и версию на Python для тех же алгоритмов (**https://github.com/dl4s/pydl4s**).

В примерах кода используется язык программирования Java и две библиотеки с открытым исходным кодом (по лицензии Apache): Apache Lucene (**http://lucene.apache.org**) и Deeplearning4j (**http://deeplearning4j.org**). Lucene является одной из наиболее широко используемых библиотек для создания поисковых систем, а Deeplearning4j на момент написания этих строк является лучшим выбором для нативной библиотеки Java для глубокого изучения. Вместе они позволят вам лег-

ко, быстро и без проблем проводить тестирование и экспериментировать с поиском и глубоким обучением.

Кроме того, многие специалисты, работающие над проектами, связанными с глубоким обучением, в настоящее время используют Python (с такими фреймворками, как TensorFlow, Keras, PyTorch и т. д.). Поэтому также предоставляется репозиторий Python, на котором размещены версии алгоритмов, подробно описанных в книге, на TensorFlow (**https://tensorflow.org**).

Исходный код в книге отформатирован `шрифтом фиксированной ширины, как этот`, чтобы отделить его от обычного текста. Во многих случаях первоначальный исходный код был переформатирован; мы добавили разрывы строк и переработали отступы, чтобы разместить доступное пространство страницы в книге. В редких случаях даже этого было недостаточно, и списки содержат маркеры продолжения строки (➥). Кроме того, комментарии в исходном коде часто удалялись из списков, когда описание кода приводится в тексте. Аннотации к коду сопровождают множество листингов, выделяя важные понятия.

Форум liveBook

Приобретение этой книги включает в себя бесплатный доступ к частному веб-форуму, организованному издательством Manning Publications, где вы можете оставлять комментарии о книге, задавать технические вопросы и получать помощь от автора и других пользователей. Чтобы получить доступ к форуму, перейдите по ссылке **https://livebook.manning.com/#!/book/deep-learning-for-search/discussion**. Вы можете узнать больше о форумах издательства и правилах поведения на странице **https://livebook.manning.com/#!/discussion**.

Обязательство Manning по отношению к нашим читателям состоит в том, чтобы обеспечить место, где может иметь место содержательный диалог между отдельными читателями и между читателями и автором. Это не обязательство какого-либо конкретного количества участия со стороны автора, чей вклад в форум остается добровольным (и неоплачиваемым). Мы предлагаем вам задавать автору сложные вопросы, чтобы его интерес не угас! Форум и архив предыдущих обсуждений будут доступны на сайте издателя, пока книга находится в печати.

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** или **www.дмк.рф** 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 этой 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.

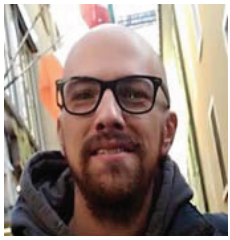
Нарушение авторских прав

Пиратство в интернете по-прежнему остается насущной проблемой. Издательства «ДМК Пресс» и Manning 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 веб-сайта, 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.

Об авторе



Томмазо Теофили – инженер-программист со страстью к открытому исходному коду и машинному обучению. Будучи участником Apache Software Foundation, он участвует в ряде проектов с открытым исходным кодом, начиная от таких тем, как поиск информации (например, Lucene и Solr), и заканчивая обработкой естественного языка и машинным переводом (включая OpenNLP, Joshua и UIMA).

В настоящее время он работает в компании Adobe, разрабатывая компоненты инфраструктуры поиска и индексации, а также исследует области обработки естественного языка, поиска информации и глубокого изучения. Он выступал с докладами о поиске и машинном обучении на конференциях, включая BerlinBuzzwords, международную конференцию International Conference on Computational Science, ApacheCon, EclipseCon и др. Вы можете найти его в Twitter (**@tteofili**).

Об иллюстрации на обложке

Рисунок на обложке книги носит название «Одеяние китайской дамы». Иллюстрация взята из книги «Коллекция платьев разных народов, древних и современных» Томаса Джефферис (четыре тома), опубликованной в Лондоне между 1757 и 1772 годом. Титульный лист гласит, что это медные гравюры ручной работы, украшенные гуммиарабиком.

Томаса Джефферис (1719–1771) называли «географом короля Георга III». Он был английским картографом, который был ведущим создателем карт своего времени. Он гравировал и печатал карты для правительственных и других государственных учреждений и выпускал обширный спектр коммерческих карт и атласов, особенно касающихся Северной Америки. Его работа в качестве картографа пробудила интерес к местному дресс-коду, блистательно представленному в этой коллекции. Он был принят в тех землях, которые он исследовал и наносил на карту. Увлечение далекими землями и путешествия ради удовольствия были относительно новым явлением в конце XVIII века, и такие коллекции, как эта, были популярны, знакомя как туристов, так и путешественников, сидящих в креслах, с жителями других стран.

Разнообразие рисунков в этом издании Джефферис ярко свидетельствует об уникальности и индивидуальности народов мира около 200 лет назад. С тех пор дресс-код изменился, а богатое в ту пору разнообразие, в зависимости от региона и страны, исчезло. Сейчас часто трудно отличить жителей одного континента от другого. Возможно, пытаясь взглянуть на это с оптимизмом, мы обменяли культурное и визуальное разнообразие на более разнообразную личную жизнь – или на более разнообразную и интересную интеллектуальную и техническую жизнь.

В то время когда трудно отличить одну компьютерную книгу от другой, издательство Manning празднует изобретательность и инициативу компьютерного бизнеса с помощью обложек книг, основанных на богатом разнообразии жизни регионов двухвековой давности, которое ожило благодаря рисункам Джефферис.

Часть I

ПОИСК ВСТРЕЧАЕТСЯ С ГЛУБОКИМ ОБУЧЕНИЕМ

Настройка поисковых систем для эффективного реагирования на потребности пользователей – непростая задача. Традиционно многие внутренние настройки и корректировки, сделанные вручную, приходилось вносить в поисковую систему, чтобы она прилично работала при реальном сборе данных. С другой стороны, глубокие нейронные сети очень хорошо подходят для изучения полезной информации об огромных объемах данных. В этой первой части книги мы начнем изучать, как можно использовать поисковую систему в сочетании с нейронной сетью, чтобы обойти некоторые общие ограничения и предоставить пользователям более совершенные возможности поиска.

Глава 1

Поиск на основе нейронных сетей

О чем идет речь в этой главе:

- деликатное введение в основы поиска;
- важные проблемы в поиске;
- почему нейронные сети могут помочь поисковым системам быть более эффективными.

Предположим, вам нужно узнать что-то о последних научных открытиях в области искусственного интеллекта. Что вы будете делать, чтобы найти информацию? Сколько времени и сил требуется, чтобы получить факты, которые вы ищете? Если вы находитесь в (огромной) библиотеке, можно спросить библиотекаря, какие книги есть по этой теме, и он, вероятно, укажет на несколько книг, о которых он знает. В идеале библиотекарь подскажет определенные главы, которые нужно искать.

Звучит довольно просто. Но библиотекарь обычно происходит из другого контекста, в отличие от вас, то есть у вас и у библиотекаря могут быть разные мнения относительно того, что является важным. В библиотеке могут быть книги на разных языках, или библиотекарь может говорить на другом языке. Информация по этой теме может быть устаревшей, учитывая, что *последняя* является довольно относительным моментом во времени, и вы не знаете, когда библиотекарь в последний раз читал что-либо об искусственном интеллекте, или регулярно ли библиотека получает публикации в этой области. Кроме того, библиотекарь может не понять ваш запрос должным образом и подумать, что вы говорите об интеллекте с точки зрения психологии[1]. Пройдет несколько повторных циклов, прежде чем вы поймете друг друга и получите нужные вам фрагменты информации.

Затем, после всего этого вы можете обнаружить, что в библиотеке нет нужной вам книги; или информация может содержаться в нескольких книгах, и вы должны прочитать их все. Как это утомительно!

Только если вы сами не библиотекарь, именно это часто происходит в наши дни, когда вы что-то ищете в интернете. Хотя мы можем рассматривать интернет как единую огромную библиотеку, существует множество разных библиотекарей,

[1] Такое случилось со мной на самом деле.

которые помогут вам найти необходимую информацию: поисковые системы. Некоторые из них являются экспертами в определенных темах; другие знают только подмножество библиотеки или лишь одну книгу.

А теперь представьте, что некто, назовем его Робби, который уже знает о библиотеке и ее посетителях, может помочь вам обмениваться данными с библиотекарем, чтобы найти то, что вы ищете. Это поможет вам быстрее получить ответы. Робби может помочь библиотекарю понять запрос посетителя, например предоставив дополнительный контекст. Робби знает, о чем обычно читает посетитель, поэтому он пропускает все книги по психологии. Также, прочитав множество книг в библиотеке, Робби лучше понимает, что является важным в области искусственного интеллекта. Было бы чрезвычайно полезно иметь таких советников, как Робби, чтобы помочь поисковым системам работать лучше и быстрее, а также помогать пользователям получать больше полезной информации.

Эта книга посвящена использованию методов из области машинного обучения, называемой *глубоким обучением*, чтобы создавать модели и алгоритмы, которые могут влиять на поведение поисковых систем, чтобы сделать их более эффективными. Алгоритмы глубокого обучения будут играть роль Робби, помогая поисковой системе обеспечить лучший опыт поиска и предоставлять более точные ответы конечным пользователям.

Важно отметить, что глубокое обучение и *искусственный интеллект* – не одно и то же. Как видно по рис. 1.1, искусственный интеллект представляет собой огромную область для исследований. Машинное обучение является лишь частью этого, а глубокое обучение, в свою очередь, является подразделом машинного обучения. В основном глубокое обучение изучает, как заставить машины «учиться», используя модель вычислений глубоких нейронных сетей.

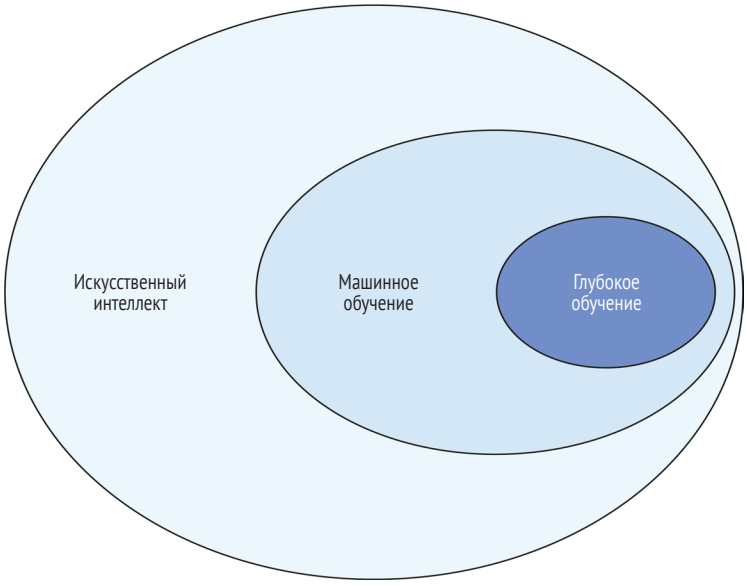


Рис. 1.1 ❖ Искусственный интеллект, машинное обучение и глубокое обучение

1.1. Нейронные сети и глубокое обучение

Цель этой книги – дать вам возможность использовать глубокое обучение в контексте поисковых систем, чтобы улучшить процесс поиска и его результаты. Даже если вы не собираетесь создавать еще одну поисковую систему в Google, вы должны научиться пользоваться методами глубокого обучения в небольших или средних поисковых системах, чтобы помочь пользователям в процессе поиска. Поиск на основе нейронных сетей должен помочь вам автоматизировать работу, которую в противном случае вам пришлось бы выполнять вручную. Например, вы узнаете, как автоматизировать извлечение синонимов из данных поисковой системы, избегая ручного редактирования файлов синонимов (глава 2). Это экономит время, повышая эффективность поиска, независимо от конкретного случая использования или области. То же самое относится и к подходящим предложениям по сопутствующему контенту (глава 6). Во многих случаях пользователи удовлетворены сочетанием простого поиска с возможностью навигации по сопутствующему контенту. Мы также рассмотрим некоторые более конкретные случаи использования, такие как поиск контента на нескольких языках (глава 7) и поиск изображений (глава 8).

Единственное требование к методам, которые мы будем обсуждать, заключается в том, что у них достаточно данных для подачи в нейронные сети. Но в общем виде трудно определить границы «достаточного количества данных». Вместо этого давайте суммируем минимальное количество документов (текст, изображения и т. д.), которые обычно необходимы для каждой проблемы, рассматриваемой в книге: *см*. табл. 1.1.

Таблица 1.1. Требования к задачам для методов поиска на базе нейронных сетей

Задача	**Минимальное количество документов (диапазон)**	**Глава**
Изучение представлений слов	1000–10 000	2, 5
Генерирование текста	10 000–100 000	3, 4
Изучение представлений документов	1000–10 000	6
Машинный перевод	10 000–100 000	7
Изучение представлений изображений	10 000–100 000	8

Обратите внимание, что не нужно строго следовать этой таблице; цифры взяты из опыта. Например, даже если в поисковой системе насчитывается менее 10 000 документов, вы все равно можете попытаться реализовать методы нейронного машинного перевода, описанные в главе 7; но вы должны принять во внимание тот факт, что получить качественные результаты может быть труднее (например, совершенные переводы).

Читая книгу, вы многое узнаете о глубоком обучении, а также обо всех необходимых основах поиска для реализации этих принципов ГО в поисковой системе. Поэтому, если вы инженер, связанный с разработкой поисковых систем, или программист, который хочет изучать поиск на базе нейронных сетей, эта книга для вас.

На данный момент вам не обязательно знать, что такое глубокое обучение или как оно работает. Вы подробнее узнаете об этом, когда мы будем рассматривать

конкретные алгоритмы один за другим, когда они станут полезными для решения определенных типов поисковых задач. А пока я начну с основных определений. Глубокое обучение – это область машинного обучения, где компьютеры способны учиться представлять и распознавать вещи постепенно, используя глубокие нейронные сети. Глубокие *искусственные нейронные сети* – это вычислительная парадигма, изначально вдохновленная тем, как мозг организован в графы нейронов (хотя мозг гораздо сложнее, чем искусственная нейронная сеть). Обычно информация поступает в нейроны во *входном слое*, затем через сеть скрытых нейронов (образуя один или несколько *скрытых слоев*), а потом выходит через нейроны в *выходном слое*. Нейронные сети также можно рассматривать как черные ящики: интеллектуальные функции, которые могут преобразовывать входные данные в результаты на основе того, чему была обучена каждая сеть. Обычная нейронная сеть имеет, по крайней мере, один входной слой, один скрытый слой и один выходной слой. Когда у сети есть более одного скрытого слоя, мы называем сеть *глубокой*. На рис. 1.2 вы видите глубокую нейронную сеть с двумя скрытыми слоями.

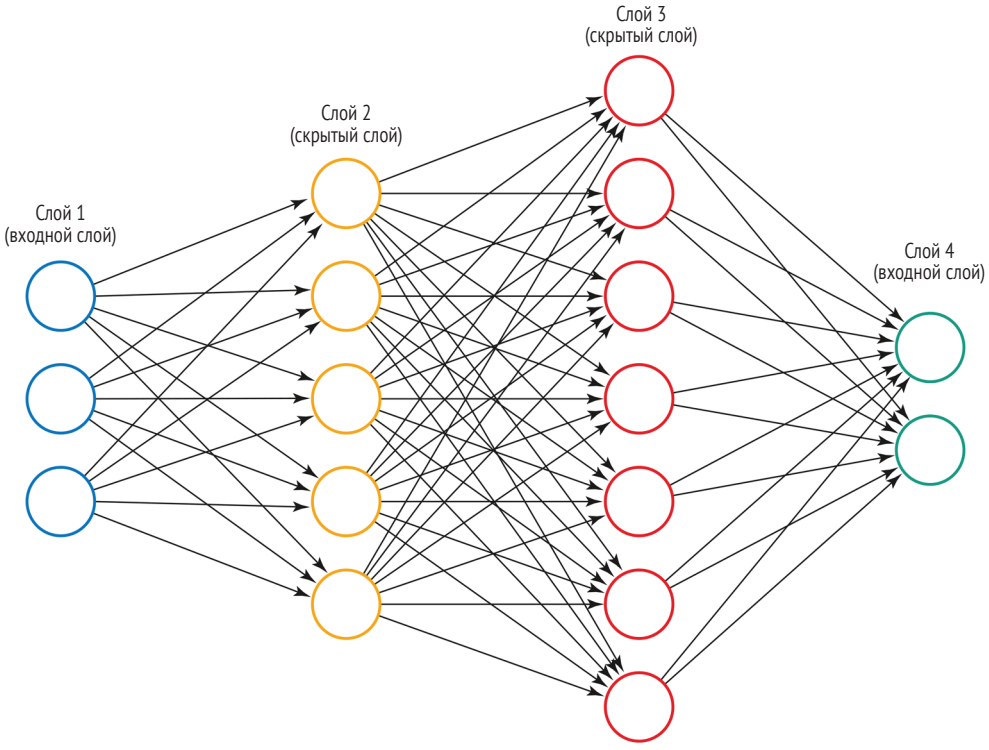


Рис. 1.2 ❖ Глубокая нейронная сеть с двумя скрытыми слоями

Прежде чем углубляться в подробности, касающиеся нейронных сетей, давайте сделаем шаг назад. Я сказал, что глубокое обучение – это подобласть машинного обучения, которое является частью более широкой области искусственного интеллекта. Но что такое машинное обучение?

1.2. Что такое машинное обучение?

Обзор основных концепций машинного обучения здесь будет полезен, прежде чем подробно рассматривать глубокое обучение и особенности поиска. Многие из концепций, которые применяются к обучению с использованием искусственных нейронных сетей, такие как *контролируемое* и *неконтролируемое* обучение, *обучение* и *прогнозирование*, берут свое начало из машинного обучения. Давайте быстро рассмотрим некоторые основные концепции машинного обучения, которые мы будем использовать в глубоком обучении (применительно к поиску) в этой книге.

Машинное обучение – это автоматизированный подход к решению проблем, основанный на алгоритмах, которые могут научиться оптимальным решениям из предыдущего опыта. Во многих случаях этот опыт проявляется в форме пар, составленных из того, что ранее наблюдалось, вместе с тем, что вы хотите, чтобы алгоритм выводил из этого. Например, алгоритм машинного обучения может быть подан текстовыми парами, где ввод – это некий текст, а вывод – категория, которая может быть использована для классификации похожих текстов. Представьте, что вы вернулись в библиотеку, но на этот раз в качестве библиотекаря. Вы купили тысячи книг и хотите разместить их на полках, чтобы люди могли с легкостью их найти. Для этого вам нужно классифицировать их так, чтобы книги, принадлежащие к одной и той же категории, помещались близко друг к другу на одной и той же книжной полке (которая, возможно, имеет небольшую метку, обозначающую категорию). Если вы можете потратить несколько часов на классификацию книг вручную, то получите опыт, необходимый вашему алгоритму. После этого вы можете обучить алгоритм на основе вашего мудрого суждения, и он будет выполнять классификацию оставшихся книг от вашего имени.

Этот тип обучения, при котором вы указываете желаемый результат, соответствующий каждому вводу, называется *контролируемым обучением*. Каждая пара, состоящая из входных данных и соответствующих им целевых результатов, называется *обучающим образцом*. В табл. 1.2 показаны некоторые из категорий, которые библиотекарь может составить вручную, чтобы помочь создать алгоритм контролируемого обучения.

Таблица 1.2. Пример данных для классификации книг

Название книги	Текст	Категории
Обработка неструктурированных текстов	Если вы читаете эту книгу, шансы, что вы программист…	OEЯ, поиск
Релевантный поиск	Заставить поисковик вести себя может быть невыносимо…	Поисковые системы, релевантность
OAuth2 в действии	Если вы разработчик программного обеспечения в интернете сегодня…	Безопасность, OAuth
Властелин колец	…	Фэнтези, романы
Lucene в действии	Lucene – это мощная поисковая библиотека Java, которая позволяет…	Lucene, поиск

Алгоритм контролируемого обучения подает данные, подобные тем, которые показаны в таблице, во время так называемой *фазы обучения*. Во время фазы обучения алгоритм машинного обучения разбивает тренировочный набор (набор

обучающих образцов) и изучает, как отобразить, например, вводимый текст в выходные категории. То, *что* изучает алгоритм машинного обучения, зависит от задачи, для которой он используется; в этом примере он используется для *классификации документов*. То, *как* алгоритм учится, зависит от того, как он сам построен. Не существует только одного алгоритма для выполнения машинного обучения. Есть различные подобласти машинного обучения, и для каждого из них существует множество различных алгоритмов.

> **ПРИМЕЧАНИЕ** Глубокое обучение – это всего лишь один из способов машинного обучения с использованием нейронных сетей. Но существует множество альтернатив, когда речь заходит о том, какой тип нейронной сети лучше всего подходит для определенной задачи. В этой книге мы будем в основном освещать темы, касающиеся машинного обучения, посредством глубокого обучения. Тем не менее мы быстро рассмотрим другие типы алгоритмов, в основном для сравнения и обоснования реальных сценариев.

После завершения фазы обучения вы обычно будете получать *модель машинного обучения*. Можно рассматривать это как артефакт, который фиксирует то, чему алгоритм научился во время обучения. Этот артефакт затем используется для выполнения *прогнозов*. Прогноз выполняется, когда модели предоставляется новый ввод без какого-либо прикрепленного желаемого вывода, и вы просите модель сообщить вам правильный вывод, основываясь на том, чему она научилась на этапе обучения. Обратите внимание, что нужно предоставить большое количество данных для обучения (не сотни, а как минимум десятки тысяч обучающих образцов), если вы хотите получить хорошие результаты при прогнозировании итогов.

В примере с классификацией книг, когда дается приведенный ниже текст, модель извлечет такие категории, как «поиск» и «Lucene»:

```
Lucene - это мощная библиотека для поиска, написанная на языке Java, которая позволяет легко
добавлять поиск в любое приложение ...
```

Это вступительные слова из книги *Lucene в действии*, *2-е изд*.

Как я уже упоминал, извлеченные категории можно использовать для размещения книг, принадлежащих к одной и той же категории, на одной и той же полке в библиотеке. Существуют ли другие способы для достижения этой цели, без предварительного предоставления учебного набора с текстами книг, помеченными по категориям? Было бы полезно, если бы вы могли найти способ измерить сходство между книгами, чтобы иметь возможность разместить похожие книги рядом друг с другом, не слишком заботясь о точном названии каждой категории. Чтобы сделать это без категорий, можно использовать методы *неконтролируемого обучения* для объединения похожих документов. В этом случае, в отличие от контролируемого обучения, алгоритм машинного обучения просматривает данные без какой-либо информации о каком-либо ожидаемом результате и извлекает шаблоны и представления данных в ходе *фазы обучения*. Во время *кластеризации* каждый фрагмент входных данных, в данном случае текст книги, преобразуется в точку, которая размещается на графике. Во время фазы обучения алгоритм кластеризации размещает точки в кластерах, предполагая, что близлежащие точки семантически похожи. После завершения обучения книги, относящиеся к одним и тем же кластерам, можно соответствующим образом подбирать и размещать на книжных полках.

В этом случае результатом неконтролируемого обучения является набор кластеров с назначенными им точками. Как и раньше, такие модели можно использовать для прогнозов, например «К какому кластеру относится эта новая книга/точка?».

Машинное обучение может помочь решить множество различных проблем, включая классификацию книг и группирование похожих текстов. До начала 2000-х годов при решении таких задач для достижения достойных результатов использовалось несколько разных методов. Затем глубокое обучение стало мейнстримом не только в исследовательских лабораториях университетов, но и отрасли. Многие проблемы машинного обучения лучше было решать с помощью глубокого обучения, поэтому оно стало более известным и более часто используемым. Успех и широкое использование глубокого обучения привели к извлечению более точных категорий книг, более точной кластеризации и множеству других улучшений.

1.3. Что глубокое обучение может сделать для поиска

Когда для решения проблем, связанных с поиском, используются глубокие искусственные нейронные сети, это поле называется поиском на базе нейронных сетей. Из этой книги вы узнаете, как составлять нейронные сети, как они работают и как их можно использовать на практике, и все это в контексте поисковых систем.

> **Поиск на базе нейронных сетей**
>
> Термин «поиск на базе нейронных сетей» является менее академической формой термина «поиск информации на базе нейронных сетей», который впервые появился во время исследовательского семинара на конференции SIGIR 2016 (**www.microsoft.com/en-us/research/event/neuir2016**), посвященной применению глубоких нейронных сетей в области поиска информации.

Вам, наверное, интересно, зачем нужен поиск на базе нейронных сетей: в конце концов, у нас уже есть хорошие поисковые системы в интернете, и нам часто удается найти то, что нам нужно. Так в чем же состоит ценность этого поиска?

Глубокие нейронные сети хорошо подходят для того, чтобы:

- обеспечить представление текстовых данных, которое фиксирует семантику слов и документов, позволяя машине определить, какие слова и документы семантически похожи;
- генерировать текст, который имеет смысл в определенном контексте: например, полезен для создания чат-ботов;
- обеспечить представления изображений, которые относятся не к пикселям, а скорее к их составным объектам. Это позволяет нам создавать эффективные системы распознавания лиц/объектов;
- эффективно выполнять машинный перевод;
- при определенных предположениях аппроксимировать любую функцию[1]. Теоретически для видов задач, которые могут выполнять глубокие нейронные сети, не существует никаких ограничений.

[1] *См.*: *Kurt Hornik, Maxwell Stinchcombe, and Halbert White*. Multilayer Feedforward Networks Are Universal Approximators // Neural Networks 2. 1989. № 5: 359–366 (**http://mng.bz/Mxg8**).

Возможно, это звучит несколько абстрактно, поэтому давайте посмотрим, какую пользу эти возможности могут принести вам как инженеру, работающему с поисковыми системами, и/или пользователю. Подумайте об основных моментах при использовании поисковых систем. Скорее всего, вы столкнетесь с такими проблемами:

- я получил плохие результаты: я нашел несколько связанных документов, но не тот, который искал;
- мне потребовалось слишком много времени, чтобы найти информацию, которую я искал (а потом я сдался);
- я должен был прочитать некоторые из предоставленных результатов, прежде чем получить представление о теме, о которой я хотел узнать;
- я искал контент на своем родном языке, но смог найти подходящие результаты только на английском;
- я искал определенное изображение, которое когда-то видел на сайте, но не смог найти его снова.

Это распространенные проблемы, и чтобы сгладить их, существуют различные решения. Но самое интересное состоит в том, что глубокие нейронные сети, если их правильно настроить, могут помочь во всех этих случаях.

С помощью алгоритмов глубокого обучения поисковая система может:

- предоставить более релевантные результаты своим конечным пользователям, повышая степень их удовлетворенности;
- выполнить поиск по двоичному содержимому, например изображениям, так же как мы ищем текст. Рассматривайте это как возможность поиска изображения с фразой «изображение леопарда, который охотится на импалу» (а вы не Google);
- предоставить контент пользователям, говорящим на разных языках, что позволяет бóльшему количеству пользователей получать доступ к данным в поисковой системе;
- как правило, становиться более чувствительной к данным, которые она обслуживает, что означает меньше шансов для запросов, которые не дают результатов.

Если вы когда-либо работали над проектированием, реализацией или настройкой поисковой системы, то наверняка сталкивались с проблемой получения решения, которое адаптируется к вашим данным. Глубокое обучение очень помогает в решении этих проблем, которые строго основаны на ваших данных, а не на фиксированных правилах или алгоритмах.

Качество результатов поиска имеет решающее значение для конечных пользователей. Есть одна вещь, которую поисковая система должна делать хорошо: выяснить, какой из возможных подходящих результатов поиска будет наиболее полезным для информационных потребностей конкретного пользователя. Хорошо ранжированные результаты поиска позволяют пользователям находить важные результаты проще и быстрее; вот почему мы уделяем большое внимание теме *релевантных результатов*. В реальной жизни это может иметь огромное значение. Согласно статье, опубликованной в *Forbes*: «Предоставляя лучшие результаты поиска, компания Netflix считает, что избегает отмененных подписок, из-за чего ее доходы будут снижаться на 1 млрд долларов в год»[1].

---

[1] *Louis Columbus*. McKinsey's State of Machine Learning and AI, 2017. 2017. July 9 (**http://mng.bz/a7KX**).

Глубокие нейронные сети могут помочь, автоматически настраивая запрос конечного пользователя за кулисами на основе прошлых запросов пользователя или на основе содержимого поисковой системы.

Сегодня люди привыкли работать с поисковыми системами для поиска изображений. Например, если вы будете искать «изображения злых львов» в Google, то получите сильно релевантные изображения. До появления глубокого обучения такие изображения должны были быть украшены *метаданными* (данными о данных), описывающими их содержание, прежде чем их помещали в поисковую систему. И эти метаданные обычно должны были набираться человеком. Глубокие нейронные сети могут абстрагировать представление изображения, которое захватывает то, что там находится, поэтому не требуется никакого вмешательства со стороны человека, чтобы поместить описание изображения в поисковую систему.

В случае с такими сценариями, как поиск в интернете (поиск по всем сайтам в интернете), пользователи могут приходить со всего мира, поэтому лучше всего, если они могут осуществлять поиск на своих родных языках. Кроме того, поисковая система может выбирать профили пользователей и возвращать результаты на их языке, даже если они используют для поиска английский язык; это распространенный сценарий для технических запросов, потому что большое количество контента создается на английском языке. Интересное применение глубоких нейронных сетей носит название *нейронного машинного перевода*. Это набор техник, которые используют глубокие нейронные сети для перевода фрагмента текста с исходного языка на другой язык.

Также интересна возможность использования глубоких нейронных сетей для изменения способа, которым поисковые системы возвращают релевантную информацию конечным пользователям. Чаще всего поисковая система будет выдавать список результатов поиска в ответ на поисковый запрос. Можно использовать методы глубокого обучения, чтобы позволить поисковой системе вернуть один фрагмент текста, который должен дать всю необходимую пользователю информацию[1]. Это избавит пользователей от просмотра каждого результата, чтобы получить все то, что им необходимо. Мы могли бы даже объединить все эти идеи и создать поисковую систему, органично предоставляющую пользователям со всего мира и текст, и изображения, которая, вместо того чтобы возвращать результаты поиска, возвращает один фрагмент текста или изображения, который нужен пользователю.

Эти приложения представляют собой примеры *поиска на базе нейронных сетей*. Как вы можете себе представить, они могут революционизировать то, как мы работаем и используем поисковые системы сегодня.

Есть много возможностей относительно того, как компьютеры могут помочь людям получить необходимую им информацию. Обсуждение нейронных сетей ведется на протяжении последних нескольких лет, но только недавно они стали так популярны; связано это с тем, что исследователи обнаружили, как сделать их намного эффективнее, чем раньше. Например, в начале 2000-х годов добавление помощи со стороны более мощных компьютеров стало ключевым шагом вперед. Чтобы воспользоваться всем потенциалом глубоких нейронных сетей, люди, ин-

[1] *Christina Lioma et al.* Deep Learning Relevance: Creating Relevant Information (As Opposed to Retrieving It). 2016. June 27 (**https://arxiv.org/pdf/1606.07660.pdf**).

тересующиеся компьютерными науками, особенно в области обработки естественного языка, компьютерного зрения и поиска информации, должны знать, как такие искусственные нейронные сети работают на практике.

Эта книга предназначена для тех, кто интересуется созданием интеллектуальных поисковых систем с помощью глубокого обучения. Это не обязательно означает, что вы будете создавать еще один поисковик Google. Это может означать, что то, что вы узнали здесь, будет использовано для проектирования и реализации эффективной поисковой системы для вашей компании или расширения вашей базы знаний, чтобы применять методы глубокого обучения в более крупных проектах, которые могут включать в себя поисковые системы.

Цель здесь состоит в том, чтобы обогатить ваши навыки, связанные с поисковыми системами и ГО, поскольку эти навыки могут быть полезны в различных контекстах. Например:

- обучение глубокой нейронной сети, чтобы научиться распознавать объекты на изображениях и использовать то, чему научилась нейронная сеть при поиске изображений;
- использование нейронных сетей для заполнения панели «связанный контент» в списке результатов поиска поисковой системы;
- обучение нейронных сетей, чтобы научиться делать пользовательский запрос более конкретным (меньше результатов поиска, но они лучше) или шире (больше результатов поиска, даже если некоторые из них могут быть менее актуальными).

1.4. Глубокое обучение: дорожная карта

Мы будем запускать наши примеры поверх программного обеспечения с открытым исходным кодом, написанного на Java, с помощью Apache Lucene (**http://lucene.apache.org**), библиотеки для поиска информации, и Deeplearning4j (**http://deeplearning4j.org**), библиотеки, используемой как фреймворк для глубокого обучения. Но мы по возможности сосредоточимся на принципах, а не на реализации, чтобы убедиться, что методы, описанные в этой книге, могут использоваться с различными технологиями и/или сценариями. На момент написания этих строк Deeplearning4j был широко используемым фреймворком для глубокого обучения в корпоративных сообществах; является частью организации Eclipse Foundation. Он также хорошо зарекомендовал себя благодаря интеграции с популярными фреймворками для реализации распределенной обработки данных, такими как Apache Spark. Полный исходный код, содержащийся в этой книге, можно найти на сайте **www.manning.com/books/deep-learning-for-search** и на GitHub по адресу **https://github.com/dl4s/dl4s**. Однако существуют и другие фреймворки для глубокого обучения. Например, TensorFlow (от компании Google) популярен среди сообществ Python и исследовательских сообществ. Почти каждый день появляются новые инструменты, поэтому я решил сосредоточиться на относительно простом в использовании фреймворке, который может быть легко интегрирован с Lucene, одной из наиболее распространенных поисковых библиотек для виртуальной машины Java. Если вы работаете с Python, то можете найти реализации большей части кода для TensorFlow, использованного в этой книге, а также некоторые инструкции по GitHub по адресу **https://github.com/dl4s/pydl4s**.

Планируя эту книгу, я решил представить главы в порядке возрастания уровня сложности, поэтому каждая глава научит определенному применению нейронных сетей для конкретной задачи поиска, поддерживаемой известными алгоритмами. Мы будем следить за современными алгоритмами глубокого обучения, но мы также осознаем, что нельзя охватить все. Цель состоит в том, чтобы обеспечить подходящие исходные данные, которые можно легко расширить, если на следующей неделе появится новый и более совершенный алгоритм на основе нейронной сети. Ключевые вещи, которые мы улучшим с помощью глубоких нейронных сетей, – это релевантность, понимание запросов, поиск изображений, машинный перевод и рекомендательные системы, работающие с документами. Не беспокойтесь, если ни один из этих терминов вам не знаком: я представлю эти задачи как есть, без какой-либо техники глубокого обучения, а затем покажу, когда и как ГО может помочь.

В первой части книги я дам обзор того, как нейронные сети могут помочь улучшить поисковые системы в целом. Сначала я сделаю это, используя приложение, в котором нейронные сети помогают поисковой системе создавать несколько версий одного и того же запроса, генерируя синонимы. Во второй части книги мы рассмотрим методы на базе глубокого обучения, чтобы сделать поисковые запросы более выразительными. Эта улучшенная выразительность сделает запросы более подходящими для целей пользователя и, таким образом, заставит поисковую систему возвращать более точные (более релевантные) результаты. Наконец, в третьей части книги мы будем работать над более сложными вещами, такими как поиск на нескольких языках и поиск изображений, и, наконец, рассмотрим аспекты производительности поисковых систем на базе нейронных сетей.

Попутно мы также сделаем паузу, чтобы рассмотреть верность и то, как измерить окончательные результаты, когда мы применяем поиск на базе нейронных сетей. Без цифр, постоянно демонстрирующих то, что мы считаем подходящим, мы далеко не уйдем. Нам нужно измерить, насколько хороши наши системы с использованием и без использования причудливых нейронных сетей.

В этой главе мы начнем с рассмотрения проблем, которые пытаются решить поисковые системы, и наиболее распространенных методов, используемых для их решения. Это исследование познакомит вас с основами анализа, загрузки и извлечения текста в поисковой системе, чтобы вы могли узнать, как запросы попадают в результаты поиска, а также с некоторыми основами решения проблемы, связанной с возвратом релевантных результатов в первую очередь. Мы также раскроем некоторые слабые стороны, присущие обычным методам поиска, что приводит к обсуждению того, для чего можно использовать глубокое обучение в контексте поиска. Затем мы посмотрим, какие задачи может помочь решить ГО и каковы практические последствия его применения в области поиска. Это поможет нарисовать реалистичную картину того, что вы можете и чего не можете ожидать от нейронного поиска в реальных ситуациях.

1.5. Получение полезной информации

Давайте начнем с изучения того, как получать результаты поиска, которые соответствуют нуждам пользователей. Это даст вам основы, необходимые для понимания того, как глубокие нейронные сети могут помочь в создании инновационных поисковых платформ.

Первый вопрос: что такое поисковая система? Это система, программа, работающая на компьютере, которую люди могут использовать для получения информации. Основная ценность поисковой системы состоит в том, что хотя она и принимает «данные», она должна предоставлять «информацию». Эта цель означает, что поисковая система должна делать все возможное, чтобы разобраться в данных, которые она получает, чтобы предоставить нечто, что может быть легко потреблено ее пользователями. Будучи пользователями, мы редко нуждаемся в большом количестве данных по определенной теме; мы часто ищем конкретную информацию, и нас бы удовлетворил только *один* ответ, а не сотни или тысячи результатов, которые нужно проверять.

Когда дело доходит до поисковых систем, большинство людей обычно думает о Google, Bing, Baidu и других крупных популярных поисковых системах, которые предоставляют доступ к огромным объемам информации, поступающей из множества разнообразных источников. Но есть также много небольших поисковых систем, которые фокусируются на контенте из определенной области или темы. Их часто называют *вертикальными поисковыми системами*, потому что они работают с ограниченным набором типов документов или тем, а не со всем контентом, который в настоящее время находится в сети. Вертикальные поисковые системы также играют важную роль, потому что часто они могут предоставить более точные результаты о «своих» данных – поскольку были адаптированы к этому конкретному контенту. Они нередко позволяют нам получать более точные результаты с более высокой верностью (например, сравните поиск академической статьи в Google и Google Scholar). (Пока мы не будем вдаваться в подробности того, что означает понятие *верность*; здесь я говорю об общей концепции верности ответа на запрос. Но верность – это также название четко определенной меры, используемой для того, чтобы оценить, насколько хороши и точны результаты информационно-поисковой системы.) На данном этапе мы не будем разграничивать размер данных и базы пользователей, потому что все концепции, которые идут далее, применимы к большинству существующих поисковых систем, независимо от их размера.

Основные обязанности поисковой системы обычно включают в себя:

- *индексирование* – эффективная загрузка и хранение данных, что позволяет быстро их извлекать;
- *запросы* – обеспечение функциональности поиска, чтобы конечный пользователь мог выполнять поиск;
- *ранжирование* – представление и ранжирование результатов в соответствии с определенными метриками для наилучшего удовлетворения информационных потребностей пользователей.

Ключевым моментом на практике также является *эффективность*. Если для получения искомой информации требуется слишком много времени, скорее всего, в следующий раз вы переключитесь на другую поисковую систему.

Но как поисковая система работает со страницами, книгами и подобными текстами? В следующих разделах вы узнаете:

- как большие куски текста разделяются на более мелкие части, чтобы поисковая система могла принять заданный запрос и быстро получить документ;
- основы того, как зафиксировать важность и релевантность результатов поиска для конкретного запроса.

Давайте начнем с основ поиска информации (индексация, запросы и ранжирование). Прежде чем заняться этим, вы должны понять, как большие потоки текстов попадают в поисковую систему; это важно, потому что это влияет на возможности быстрого поиска поисковой системы и предоставления важных результатов.

1.5.1. Текст, токены, термы и основы поиска

Поставьте себя на место библиотекаря, который только что получил запрос на книги, связанные с определенной темой. Как узнать, что одна из книг содержит информацию по определенной теме? Как узнать, что книга даже содержит определенное слово?

Извлечение категорий, к которым принадлежит определенная книга (темы высокого уровня, такие как «искусственный интеллект» и «глубокое обучение»), отличается от извлечения всех слов, содержащихся в книге. Например, категории облегчают поиск книги об искусственном интеллекте для новичка, поскольку предварительного знания специфических для искусственного интеллекта методов или авторов не требуется. Пользователь зайдет на сайт поисковой системы и начнет просматривать существующие категории и искать что-то, что достаточно близко к теме искусственного интеллекта. С другой стороны, для эксперта по искусственному интеллекту знание того, содержит ли книга слова *градиентный спуск* или *метод обратного распространения ошибки*, позволяет находить результаты, которые содержат более детальную информацию об определенных методах или проблемах в области ИИ.

Людям, как правило, трудно вспомнить все слова, содержащиеся в книге, хотя мы легко можем определить тему книги, прочитав несколько абзацев из нее или даже посмотрев на пролог или предисловие. Компьютеры, как правило, ведут себя иначе. Они могут легко хранить большие объемы текста и «запоминать» все слова, содержащиеся на миллионах страниц, чтобы их можно было использовать при поиске; с другой стороны, они не так хорошо извлекают информацию, которая скрыта, разбросана или не сформулирована непосредственно в данном фрагменте текста, например к какой категории относится книга. Например, в книге о нейронных сетях никогда не упоминается «искусственный интеллект» (хотя это, вероятно, будет относиться к машинному обучению). Но это все равно относится к широкой категории «книг об искусственном интеллекте».

Давайте сначала посмотрим на задачу, которую компьютеры уже могут выполнять: извлечение и хранение фрагментов текста (также известных как *термы*) из потоков текста. Можно рассматривать этот процесс, который носит название *анализ текста*, как разбиение текста книги на все составляющие его слова. Представьте себе ленту, на которой содержимое книги написано потоком, и машину (алгоритм анализа текста), в которую вы вставляете такую ленту в качестве ввода. Вы получаете множество фрагментов такой ленты в качестве результатов, и каждый из этих выходных фрагментов содержит слово, или предложение, или именную фразу (например, «искусственный интеллект»); вы можете понять, что некоторые слова, записанные на ленте ввода, были съедены машиной и не были выведены в какой-либо форме.

Поскольку последние единицы, которые должны быть созданы алгоритмом анализа текста, могут быть словами, но также могут быть группой слов, или предложений, или даже частями слов, мы называем эти фрагменты термами. Можно

рассматривать терм как основную единицу, которую поисковая система использует для хранения данных и, следовательно, их извлечения.

Это основа одной из самых фундаментальных форм поиска – *поиска по ключевым словам*: пользователь вводит набор слов и ожидает, что поисковая система выдаст все документы, которые содержат некоторые или все эти термы. Так начинался поиск в интернете десятилетия назад. Хотя сегодня многие поисковые системы намного умнее, немало пользователей продолжает составлять запросы на основе ключевых слов, которые, как они ожидают, будут содержаться в результатах поиска. Это то, что вы увидите сейчас: как текст, введенный пользователем в поле поиска, заставляет поисковую систему возвращать результаты. *Запрос* – это то, что мы называем текстом, который вводит пользователь, чтобы что-то найти. Хотя запрос – это просто текст, он передает и кодирует то, что нужно пользователю, и то, как пользователь выражает эту, возможно, общую или абстрактную потребность (например, «Я хочу узнать о последних и самых больших исследованиях в области искусственного интеллекта») в способ, который является кратким, но все же описательным (например, «последнее исследование в области искусственного интеллекта», как показано на рис. 1.3).

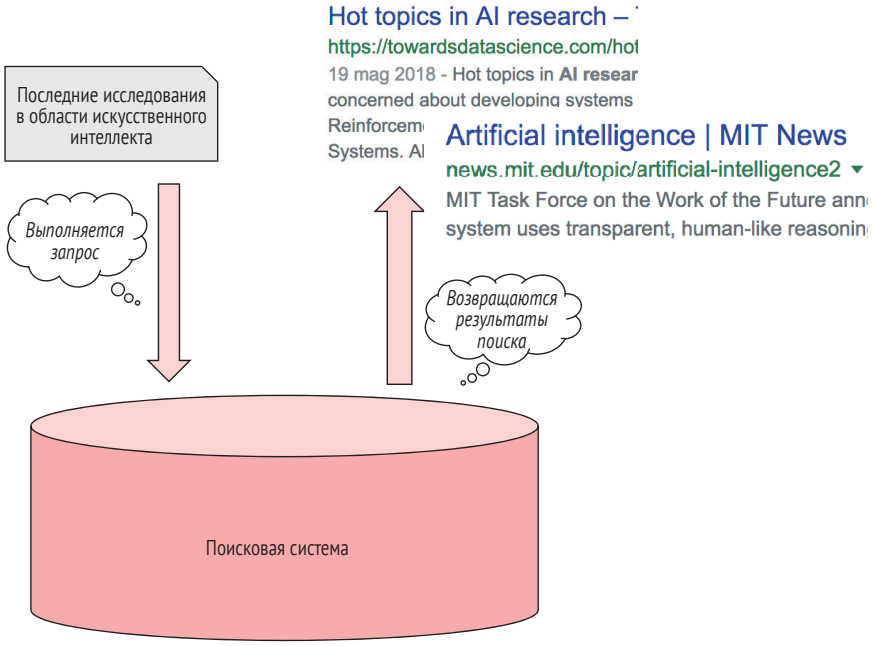


Рис. 1.3 ❖ Поиск и получение результатов

Если, будучи пользователем, вы хотите найти документы, содержащие слово «поиск», как поисковая система будет возвращать такие документы? Не слишком умный способ сделать это – просмотреть содержимое каждого документа с самого начала и сканировать его, пока поисковая система не найдет совпадение. Но выполнять такие проверки текста для каждого запроса очень затратно, особенно когда речь идет о большом количестве больших документов:

- многие документы могут не содержать слова «поиск»; поэтому их сканирование было бы напрасной тратой вычислительных ресурсов;
- даже если документ содержит слово «поиск», это слово может встречаться ближе к концу документа, что требует от поисковой системы «прочитать» все предыдущие слова, прежде чем найти совпадение для слова «поиск».

У вас есть *совпадение* или *попадание*, когда в результате поиска найден один или несколько *термов*, являющихся частью запроса.

Вам нужно найти способ быстро вычислить этот этап поиска. Одним из фундаментальных методов для достижения данной цели является разбиение предложений типа «мне нравятся поисковые системы» на более мелкие единицы: в данном случае [«мне», «нравятся», «поисковые», «системы»]. Это обязательное условие для эффективных механизмов хранения, называемых *инвертированными индексами*, о которых мы поговорим в следующем разделе. Программа анализа текста часто организована в виде конвейера: цепочки компонентов, каждый из которых принимает вывод предыдущего компонента в качестве ввода. Такие конвейеры обычно состоят из строительных блоков двух типов:

- *токенизаторы* – компоненты, которые разбивают поток текста на слова, фразы, символы или другие единицы, называемые *токенами*;
- *фильтры токенов* – компоненты, принимающие поток токенов (от токенизатора или другого фильтра) и могущие изменять, удалять или добавлять новые токены.

Вывод таких конвейеров анализа текста представляет собой последовательность следующих друг за другом термов, как показано на рис. 1.4.

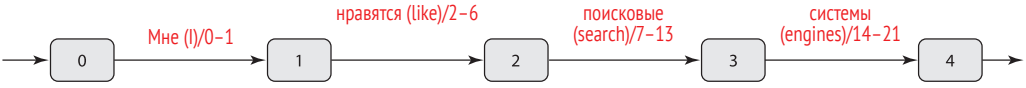


Рис. 1.4 ❖ Получение слов «мне нравятся поисковые системы» с использованием простого конвейера анализа текста

Теперь вы знаете, что анализ текста полезен по соображениям производительности для создания быстрых поисковых систем. Другим не менее важным аспектом является то, что он контролирует соответствие запросов и текста, которые будут помещены в индекс. Часто конвейеры анализа текста используются для фильтрации токенов, которые не считаются полезными или необходимыми для поисковой системы. Например, избегать хранения распространенных термов, таких как статьи или предлоги, в поисковой системе является обычной практикой, поскольку эти слова существуют в большинстве текстовых документов на таких языках, как английский, и обычно вам не нужно, чтобы запрос возвращал все в поисковой системе: это не принесло бы много пользы пользователю. В подобных случаях можно создать фильтр, отвечающий за удаление таких токенов, как «the», «a», «an», «of», «in» и т. д., позволяя всем остальным токенам вытекать по мере того, как токенизатор будет производить их. В этом упрощенном примере:

- токенизатор будет разделять токены каждый раз, когда будет встречать символ пробела;
- фильтр токенов удалит токены, которые соответствуют определенному черному списку (также известному как список *стоп-слов*).

В реальной жизни, особенно при первоначальной настройке поисковой системы, принято создавать несколько различных алгоритмов анализа текста и проверять их на данных, которые вы хотите поместить в поисковую систему. Это позволяет визуализировать, как контент будет обрабатываться такими алгоритмами, например какие токены генерируются, какие в конечном итоге отфильтровываются и т. д. Вы создали эту цепочку анализа текста (также называемую *анализатором*) и хотите убедиться, что она работает должным образом и фильтрует статьи, предлоги и т. д. Давайте попробуем передать первый фрагмент текста упрощенному анализатору и отправить предложение «the brown fox jumped over the lazy dog» (бурая лисица перепрыгнула через ленивого пса) в конвейер. Вы ожидаете, что статьи будут удалены. Сгенерированный выходной поток будет выглядеть так, как показано на рис. 1.5.

Рис. 1.5 ❖ Схема токенов

В полученном потоке токенов, как и ожидалось, «токены» удалены; это можно увидеть из пунктирных стрелок в начале графа и между узлами «over» и «lazy». Числа рядом с токенами обозначают начальную и конечную позиции (в количестве символов) каждого токена. Важным моментом этого примера является то, что запрос «the» не будет соответствовать, потому что анализатор удалил все подобные токены, и они не будут частью содержимого поисковой системы. В реальной жизни конвейеры анализа текста часто являются более сложными; некоторые из них вы увидите в следующих главах. Теперь, когда вы познакомились с анализом текста, давайте посмотрим, как поисковые системы хранят текст (и термы), чтобы они были запрошены конечными пользователями.

Индексация

Хотя поисковая система должна разбивать текст на термы для быстрого поиска, конечные пользователи ожидают, что результаты поиска будут представлены в виде единого блока: документа. Представьте себе результаты поиска в Google. Если вы ищете слово «книги», то получите список результатов, каждый из которых состоит из заголовка, ссылки, фрагмента текста и т. д. Каждый из этих результатов содержит слово «книги», но вы видите документ, который содержит гораздо больше информации и контекста, чем просто текстовый фрагмент, где это слово совпало. На практике токены, полученные в результате анализа текста, хранятся со ссылкой на оригинальный фрагмент текста, которому они принадлежат.

Эта связь между термом и документом позволяет:

- подобрать ключевое слово или поисковый терм из запроса;
- вернуть указанный исходный текст в качестве результата поиска.

Весь этот процесс анализа потоков текста и хранения результирующих термов (вместе со ссылочными документами) в поисковой системе обычно называют *индексацией*.

Причиной такой формулировки является то, что термы хранятся в *инвертированном индексе*: структуре данных, которая отображает терм в текст, в котором он

изначально содержался. Вероятно, самый простой способ взглянуть на это – это аналитический указатель реальной книги, где каждое слово указывает на страницы, на которых оно упомянуто; в случае с поисковой системой слова – это термы, а страницы – оригинальные части текста.

Отныне мы будем обозначать фрагменты текста для индексирования (страницы, книги) как *документы*. Чтобы наглядно представлять себе, что происходит с документами после индексации, давайте предположим, что у вас есть два очень похожих документа:

- «the brown fox jumped over the lazy dog» (бурая лисица перепрыгнула через ленивого пса) (документ 1);
- «a quick brown fox jumps over the lazy dog» (быстрая бурая лисица перепрыгивает через ленивого пса) (документ 2).

Предполагая, что вы используете алгоритм анализа текста, определенный ранее (токенизация пробелов со стоп-словами «a», «an» и «the»), в табл. 1.3 приводится подходящая аппроксимация инвертированного индекса, содержащего такие документы.

Таблица 1.3. Инвертированный индекс

Терм	Идентификаторы документа
brown	1, 2
fox	1, 2
jumped	1
over	1, 2
lazy	1, 2
dog	1, 2
quick	2
jumps	1

Как видно, здесь нет записи для терма «the», потому что фильтр токенов на основе стоп-слов удалил подобные токены. В таблице можно найти словарь термов в первом столбце и *постинг-лист* – набор идентификаторов документов, – связанный с каждым термом в каждой строке. При использовании инвертированных индексов поиск документов, содержащих данный терм, выполняется очень быстро: поисковая система выбирает инвертированный индекс, ищет запись для поискового терма и в конечном итоге извлекает документы, содержащиеся в постинг-листе. В примере индекса, если вы ищете терм «quick» (быстрая), инвертированный индекс вернет документ 2, просмотрев постинг-лист, соответствующий терму «quick». Мы только что рассмотрели быстрый пример индексации текста в поисковую систему.

Давайте подумаем о шагах по индексации книги. Книга состоит из страниц, основного содержимого, но у нее также есть название, автор, редактор, год публикации и т. д. Вы не можете использовать один и тот же конвейер анализа текста для всего, и вам бы не хотелось удалять слова «the» или «an» из названия книги. Пользователь, знающий название книги, должен найти ее с помощью точного сопоставления! Если цепочка анализа текста удаляет предлог «in» (в) из названия книги *Tika in Action* («Тика в действии»), запрос «Tika in Action» не позволит

вам найти ее. С другой стороны, вы можете избежать хранения таких токенов для содержания книги, поэтому у вас есть конвейер анализа текста, который более агрессивен при фильтрации нежелательных термов. Если цепочка анализа текста удаляет слова «in» и «the» из названия книги *Living in the Information Age* («Жизнь в век информации»), это не должно быть проблематичным: очень маловероятно, что пользователь будет искать «Living in the Information Age», но он может искать «information age». В этом случае потери информации практически нет, но вы получаете преимущество хранения небольших текстов и, что более важно, повышения релевантности (об этом мы поговорим в следующем разделе). Обычный подход в реальной жизни состоит в том, чтобы иметь несколько инвертированных индексов, которые обращаются к индексации различных частей документа, и все это происходит в одной поисковой системе.

Поиск

Теперь, когда у нас есть контент, проиндексированный в поисковой системе, мы рассмотрим поиск. Традиционно первые поисковые системы позволяли пользователям выполнять поиск с помощью определенных термов, также известных как *ключевые слова*, и в конечном итоге логических операторов, которые позволяют пользователям определять, какие термы *должны* совпадать, *не должны* совпадать или *могут* совпадать в результатах поиска. Наиболее часто терм в запросе *должен* совпадать, но это не обязательно. Если вам нужны результаты поиска, которые должны содержать такой терм, вы должны добавить соответствующий оператор: например, поставить знак + перед термом. Запрос типа «deep + learning for search» требует результатов, которые содержат и слово «deep», и слово «learning», а еще могут содержать слова «for» и «search». Также принято разрешать пользователям указывать, что им нужны целые фразы, а не отдельные термы. Это позволяет им искать точную последовательность слов вместо отдельных термов. Предыдущий запрос можно перефразировать как «“deep learning” for search», чтобы возвращать результаты поиска, которые должны содержать последовательность «deep learning» и, необязательно, термы «for» и «search».

Это может звучать удивительно, но анализ текста также важен на этапе поиска. Предположим, вы хотите найти книгу *Deep Learning for Search* поверх данных, которые вы только что проиндексировали; при условии что у вас есть веб-интерфейс, вы, вероятно, наберете запрос типа «deep learning for search». Задача на этом этапе поиска состоит в том, чтобы сделать возможным поиск нужной книги. Первое, что находится между пользователем и пользовательским интерфейсом классической поисковой системы, – это *синтаксический анализатор запросов*.

Анализатор запросов отвечает за преобразование текста поискового запроса, введенного пользователем, в набор операторов, которые указывают, какие термы должна искать поисковая система и как их использовать при поиске совпадения в инвертированных индексах. В предыдущих примерах запросов анализатор отвечал за понимание символов + и ”. Еще один широко распространенный синтаксис позволяет помещать логические операторы в термы запроса: «deep AND learning». В этом случае анализатор запросов придаст особое значение оператору «AND»: термы слева и справа от него являются обязательными. Синтаксический анализатор запросов можно рассматривать как функцию, которая берет некий текст и выводит набор ограничений для применения к базовому инвертированному индексу (индексам) для поиска результатов. Давайте снова возьмем при-

мер запроса типа «latest research in artificial intelligence» (последние исследования в области искусственного интеллекта). Умный анализатор запросов будет создавать операторы, отражающие семантику слов; например, вместо двух операторов для слов «artificial» и «intelligence» он должен создать только один оператор для «artificial intelligence». Кроме того, вероятно, терм «latest» не должен совпадать; вам не нужны результаты, содержащие слово «latest»; вместо этого вам нужно получить результаты, которые были «созданы» недавно. Таким образом, хороший анализатор запросов преобразует терм «latest» в оператор, который можно выразить, например, как «созданный между сегодняшним днем и двумя месяцами ранее» на естественном языке. Механизм запросов будет кодировать такого оператора таким образом, чтобы его было легче обработать с помощью компьютера, например `created < today() AND created > (today() - 60days)`; *см.* рис. 1.6.

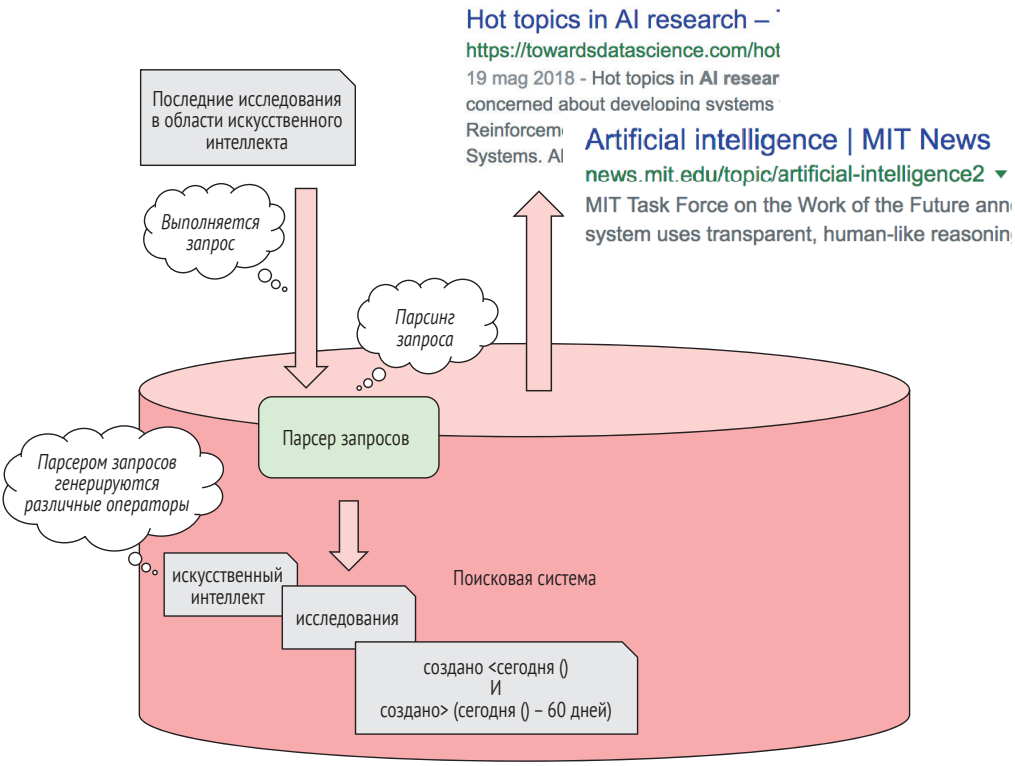


Рис. 1.6 ❖ Парсинг запросов

Во время индексации конвейер анализа текста используется для разделения входного текста на термы, которые должны храниться в индексе; это также называется *анализом текста во время индексации*. Точно так же анализ текста может применяться во время поиска по запросу, чтобы разбить строку запроса на термы, поэтому это называется *анализом текста во время поиска*. Документ извлекается поисковой системой, когда термы времени поиска совпадают с термом в инвертированном индексе, на который ссылается этот документ.

На рис. 1.7 показан анализ во время индексации, который используется для разделения текста документа на термы. Они попадают в индекс и все ссылаются на *документ 1*. Анализ во время индексации состоит из токенизатора пробелов и двух фильтров: первый используется для удаления нежелательных стоп-слов (например, «the»), а второй – для преобразования всех термов в нижний регистр (например, «Fox» преобразуется в «fox»). В правом верхнем углу запрос «lazy foxes» (ленивые лисицы) передается в анализ во время поиска, который разбивает токены с помощью токенизатора пробельных символов, но выполняет фильтрацию с использованием фильтра строчных букв и стемминг-фильтра. Стемминг-фильтр преобразует термы, приводя флективные или производные слова к их корневой форме; это означает удаление множественных суффиксов, -ing формы в глаголах и т. д. В этом случае «foxes» превращаются в «fox».

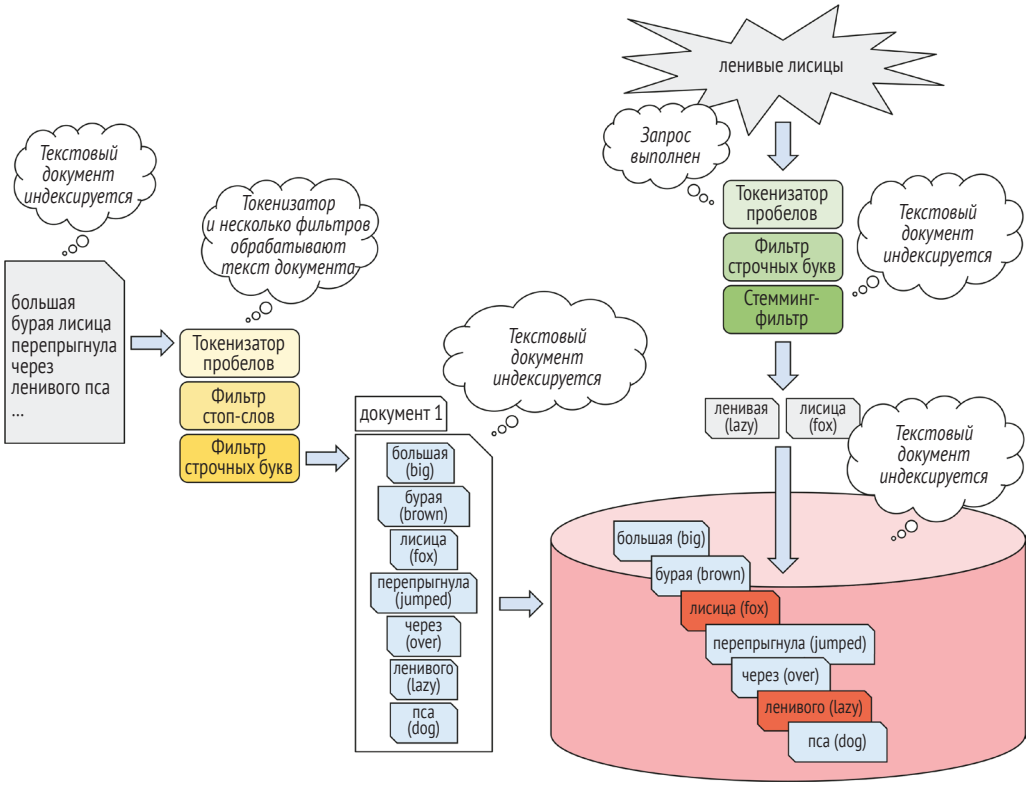


Рис. 1.7 ❖ Индекс, анализ времени поиска и сопоставление термов

Привычный способ убедиться, что конвейеры индексирования и анализа текста работают должным образом, состоит в следующем:

1) взять образец контента;
2) передать контент в цепочку анализа текста во время индексации;
3) взять пример запроса;
4) передать запрос в цепочку анализа текста во время поиска;
5) проверить полученные термы на предмет соответствия.

Например, во время индексации обычно используются фильтры стоп-слов, потому что выполнение фильтрации не повлияет на производительность на этапе поиска. Но возможно иметь и другие фильтры на этапах индексирования или поиска. Имея цепочки анализа текста во время индексации и поиска и синтаксический анализ запросов, мы можем увидеть, как работает процесс получения результатов поиска.

Вы изучили один из базовых методов, лежащих в основе каждой поисковой системы: анализ текста (токенизация и фильтрация) позволяет системе разбивать текст на термы, которые, как вы ожидаете, пользователи будут вводить во время запроса, и помещать их в структуру данных под названием инвертированный индекс, который обеспечивает эффективное хранение (по пространству) и поиск (по времени). Однако, будучи пользователями, мы не хотим просматривать все результаты поиска, поэтому нужна поисковая система, чтобы она сообщала нам, какие из них должны быть наиболее подходящими. Теперь вам, наверное, интересно, что значит *наиболее подходящие*? Можно ли измерить, насколько полезна информация, учитывая наши запросы? Ответ – да: мы называем это *релевантностью*. Точное ранжирование результатов поиска – одна из самых важных задач, которую должен выполнить поисковик. В следующем разделе мы кратко рассмотрим, как решить проблему релевантности.

1.5.2. Релевантность прежде всего

Теперь вы знаете, как поисковые системы получают документ по заданному запросу. В этом разделе вы узнаете, как поисковые системы ранжируют результаты поиска, чтобы самые важные результаты возвращались первыми. Это даст вам четкое представление о работе обычных поисковых систем.

Релевантность является ключевым понятием в поиске; это мера того, насколько важен документ для определенного поискового запроса. Будучи людьми, нам часто легко определить, почему некоторые документы более актуальны, чем другие, в отношении запроса. Таким образом, теоретически мы могли бы попытаться извлечь набор правил, отражающих наши знания о ранжировании важности документа. Но на практике такое упражнение, вероятно, потерпит неудачу:

- объем информации, которая у нас есть, не позволяет нам извлечь набор правил, применимых к большинству документов;
- документы в поисковой системе со временем сильно меняются, и очень важно постоянно корректировать правила соответствующим образом;
- документы в поисковой системе могут принадлежать разным доменам (например, в поиске по сети), и невозможно найти хороший набор правил, который работает для всех типов информации.

Одной из центральных тем в области поиска информации является определение модели, для которой не требуется специалист для извлечения таких правил. Подобная *модель поиска* должна как можно точнее охватывать понятие релевантности. Учитывая набор результатов поиска, поисковая модель будет ранжировать *каждый* из них: чем более релевантен результат, тем выше его оценка.

В большинстве случаев, будучи специалистом по разработке поисковых систем, вы не получите идеальных результатов, просто выбрав поисковую модель; релевантность – это капризная бестия! В реальных сценариях вам, возможно, придется постоянно корректировать свои конвейеры анализа текста, а также модель

поиска и, возможно, выполнять тонкую настройку внутренних компонентов поисковой системы. Но модели поиска очень помогают, предоставляя надежную основу для получения хорошей релевантности.

1.5.3. Классические модели поиска

Вероятно, одной из наиболее часто используемых моделей поиска информации является модель векторного пространства[1]. В этой модели каждый документ и запрос представлен в виде вектора. Можно рассматривать вектор как стрелку в координатной плоскости; каждая стрелка в векторной модели может обозначать запрос или документ. Чем ближе две стрелки, тем больше они похожи (*см.* рис. 1.8); направление каждой стрелки определяется термами, которые образуют запрос/документ.

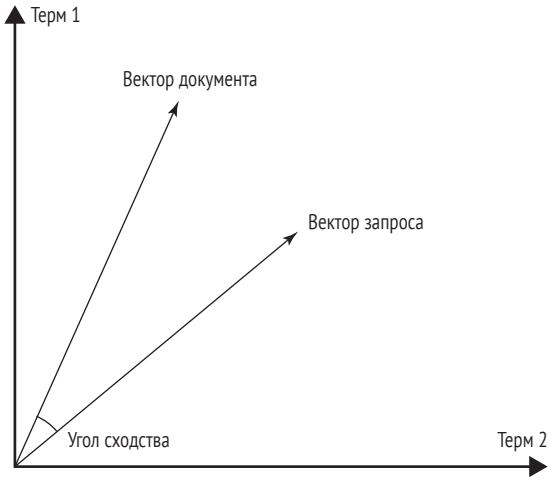


Рис. 1.8 ❖ Сходство между векторами документа и запроса согласно модели векторного пространства

В таком векторном представлении каждый терм связан с *весом*: действительным числом, которое указывает, насколько этот терм важен в этом документе/запросе по отношению к остальным документам в поисковой системе. Такой вес можно рассчитать различными способами. На этом этапе мы не будем вдаваться в подробности того, как это делается. Я упомяну, что наиболее распространенный алгоритм называется TF-IDF (*term frequency – inverse document frequency – частота терма – обратная частота документа*). Основная идея, лежащая в основе TF-IDF, заключается в том, что чем чаще терм появляется в одном документе (частота терма, или TF), тем он важнее. В то же время он устанавливает, что чем более распространен терм среди всех документов, тем менее он важен (обратная частота документа, или IDF). Таким образом, в векторной модели результаты поиска ранжируются по вектору запроса, поэтому документы отображаются выше в списке

[1] *См.*: *G. Salton, A. Wong, and C. S. Yang*. A vector space model for automatic indexing // Communications of the ACM 18. 1975. № 11: 613–620 (**http://mng.bz/gNxG**).

результатов (получают более высокий ранг/оценку), если они находятся ближе к такому вектору.

В то время как векторная модель – это информационно-поисковая модель на базе линейной алгебры, с годами появились альтернативные подходы, основанные на вероятностных моделях релевантности. Вместо того чтобы вычислять, насколько близко находятся документ и вектор запроса, вероятностная модель ранжирует результаты поиска на основе оценки вероятности того, что документ является релевантным к определенному запросу. Одна из самых распространенных функций ранжирования для таких моделей – это *Okapi BM25*. Мы не будем углубляться в детали, но она показала хорошие результаты, особенно в не очень длинных текстах.

1.5.4. Точность и полнота

Мы рассмотрим, как поиск на базе нейронных сетей может помочь в определении релевантности в следующих главах, но для начала нам нужно измерить релевантность! Стандартный способ измерения того, насколько хорошо работает информационно-поисковая система, – это расчет ее точности и повторного вызова. *Точность* – это доля найденных документов, которые являются релевантными. Если система имеет высокую точность, пользователи в основном находят результаты поиска в верхней части списка результатов. *Полнота* – это доля релевантных документов, которые были найдены. Если у системы хорошая полнота, пользователи найдут все релевантные для них результаты в результатах поиска, хотя не все они могут быть в числе топовых.

Как вы, возможно, заметили, для измерения точности и полноты требуется, чтобы кто-то судил о том, насколько релевантны результаты поиска. В небольших случаях это решаемая задача; но требуемое усилие делает это едва выполнимым для огромных коллекций документов. Возможность измерить эффективность поисковых систем заключается в использовании общедоступных наборов данных для поиска информации, например из серий конференций TREC[1] Национального института стандартов и технологий (NIST), в которых содержится множество ранжированных запросов, чтобы использовать их для проверки точности и полноты.

В этом разделе вы узнали некоторые основы классических моделей поиска информации, таких как векторная модель и вероятностные модели. Теперь мы рассмотрим распространенные проблемы, которые влияют на поисковые системы. В остальной части книги обсудим, как исправить их с помощью глубокого обучения.

1.6. Нерешенные проблемы

Мы более подробно рассмотрели, как работает поисковая система, в частности как она стремится извлекать информацию, соответствующую потребностям конечного пользователя. Но давайте сделаем шаг назад и попытаемся увидеть проблему с точки зрения того, как мы, пользователи, каждый день пользуемся поисковыми системами. Мы рассмотрим некоторые проблемы, которые остаются нерешенными во многих поисковых сценариях, чтобы лучше понять, какие задачи мы можем надеяться решить с помощью глубокого обучения.

[1] **http://trec.nist.gov/data.html**.

Заполнение пробела в знаниях, в отличие от поиска информации, – несколько более сложная тема. Давайте снова возьмем в качестве примера поход в библиотеку, потому что вы хотите узнать больше об интересных недавних исследованиях в области искусственного интеллекта. Как только вы встречаетесь с библиотекарем, у вас возникает проблема: как заставить библиотекаря четко понять, что вам нужно и что было бы полезно для вас?

Хотя это звучит просто, полезность информации вряд ли объективна. Она скорее субъективна и основана на контексте и мнении. Вы можете предположить, что у библиотекаря достаточно знаний и опыта, для того чтобы вы получали то, что вам нужно. В реальной жизни вы, вероятно, поговорите с библиотекарем, чтобы представиться и поделиться информацией о вашем прошлом и о том, зачем вам нужно что-либо; что позволило бы библиотекарю использовать такой контекст, чтобы:

- исключить некоторые книги, прежде чем пытаться искать;
- отказаться от некоторых книг, найдя их;
- выполнить явный поиск в одной или нескольких областях, которые имеют более близкое отношение к вашему контексту (например, в научных или отраслевых источниках).

Вы сможете дать отзыв о книгах, переданных вам библиотекарем, позже, хотя иногда вы можете выразить озабоченность, основываясь на прошлом опыте (например, вам не нравятся книги, написанные определенным автором, поэтому вы советуете библиотекарю явно не учитывать их). И контекст, и мнение могут значительно различаться и, следовательно, влияют на актуальность информации с течением времени среди разных людей. Как библиотекарь справляется с этим несоответствием?

Вы как пользователь можете не знать библиотекаря или, по крайней мере, недостаточно хорошо понимать его контекст. Подоплека и мнения библиотекаря важны, потому что они влияют на результаты, которые вы получаете. Поэтому чем лучше вы понимаете библиотекаря, тем быстрее вы получите свою информацию. Так что вам нужно знать своего библиотекаря, чтобы получить нужные результаты!

Что, если библиотекарь даст вам книгу о «методах глубокого обучения» в ответ на ваш первый запрос об «искусственном интеллекте»? Если вы не знаете предмет, вам нужно сделать второй запрос о «введении в глубокое обучение» и есть ли по этой теме подходящая книга в библиотеке. Этот процесс может повторяться несколько раз; главное – это понять, что информация течет постепенно: вы не загружаете вещи в свой мозг, как это делают персонажи в фильме «*Матрица*». Вместо этого, если вы хотите что-то узнать об искусственном интеллекте, вы можете осознать, что сначала вам нужно познакомиться с глубоким обучением, а для того чтобы сделать это, вы обнаружите, что вам нужно прочитать книги о математическом анализе и линейной алгебре и т. д. Другими словами, вы не знаете всего, что вам нужно, когда спрашиваете впервые.

Подводя итог, можно сказать, что процесс получения информации, которую вы ищете у библиотекаря, имеет некоторые недостатки, вызванные следующими ситуациями:

- библиотекарь не знает вас;
- вы не знаете библиотекаря;
- вам может понадобиться несколько циклов, чтобы получить все, что нужно.

Важно идентифицировать эти проблемы, потому что мы хотим использовать глубокие нейронные сети, чтобы они помогли нам создавать более совершенные поисковые системы, которые можно было бы легко использовать, – и мы хотели бы, чтобы глубокое обучение помогло решать эти проблемы. Понимание данных проблем является первым шагом к их решению.

1.7. Открываем черный ящик поисковой системы

Теперь давайте попытаемся понять, сколько из того, что делает поисковая система, видят пользователи. Важной проблемой при создании эффективных поисковых запросов является то, какой язык запросов вы используете. Несколько лет назад вы вводили одно или несколько ключевых слов в поле поиска, чтобы выполнить запрос. Сегодня технология эволюционировала до такой степени, что вы можете вводить запросы на естественном языке. Некоторые поисковые системы индексируют документы на нескольких языках (например, для поиска в сети) и допускают последующие запросы. Если вы ищете одно и то же, но выражаете это с помощью несколько разных запросов в поисковой системе, такой как Google, то увидите поразительно разные результаты.

Давайте проведем небольшой эксперимент, чтобы увидеть, как изменяются результаты поиска, когда один и тот же запрос выражается по-разному. Если бы вы разговаривали с человеком и задавали один и тот же вопрос по-разному, то ожидали бы всегда получать один и тот же ответ. Например, если вы спросите кого-то: «Каковы “последние достижения в искусственном интеллекте”, по вашему мнению?», то получите ответ на основе его мнения. Если вы спросите того же человека: «Каковы, по вашему мнению, “последние достижения в области искусственного интеллекта”?», то, вероятно, получите точно такой же ответ или тот, который семантически эквивалентен.

Сегодня в случае с поисковыми системами часто бывает не так. В табл. 1.4 представлены результаты поиска «последних достижений в области искусственного интеллекта» и некоторых вариантов в Google.

Таблица 1.4. Сравнение похожих запросов

Запрос	Заголовок первого результата
Latest breakthroughs in artificial intelligence	Academic papers for “latest breakthroughs in artificial intelligence” (Google Scholar)
Latest advancements in artificial intelligence	Google advancements artificial intelligence push with 2 top hires
Latest advancements on artificial intelligence	Images related to “latest advancements on artificial intelligence” (Google Images)
Latest breakthroughs in AI	Artificial Intelligence News—ScienceDaily
Più recenti sviluppi di ricerca sull’intelligenza artificiale	Intelligenza Artificiale (Wikipedia)

Хотя первый результат первого запроса неудивителен, изменение терма «breakthroughs» (прорывы) на один из его синонимов («advancements») приводит к другому результату, который, по-видимому, предполагает, что поисковая система по-разному понимает необходимую информацию: вы не смотрели, как Google

улучшает искусственный интеллект! Третий запрос дает удивительный результат: изображения! У нас нет реального объяснения этому. Изменение сочетания «искусственный интеллект» на его аббревиатуру «ИИ» (AI) приводит к другому, но по-прежнему релевантному результату. А когда вы используете перевод исходного запроса на итальянский, то получаете совершенно иной результат по отношению к запросу на английском языке: страницу Википедии об искусственном интеллекте. Это кажется обобщенным, учитывая, например, что Google Scholar индексирует научные статьи на разных языках.

Рейтинги в поисковых системах могут значительно различаться, как и мнения пользователей. Хотя специалист по разработке поисковых систем может оптимизировать ранжирование, чтобы отвечать на ряд заданных запросов, трудно настроить его для, возможно, десятков или сотен подобных запросов. Поэтому в реальной жизни мы не настраиваем ранжирование результатов поиска вручную; это будет почти невозможно и вряд ли приведет к хорошему ранжированию в общем.

Часто поиск выполняется методом проб и ошибок: вы запускаете первоначальный запрос и получаете слишком много результатов; вы выполняете второй запрос и все равно получаете слишком много результатов; а третий запрос может вернуть банальные результаты, которые вам не интересны. Выражение информационной потребности с помощью поискового запроса не является тривиальной задачей. Часто все заканчивается тем, что выполняете кучу запросов только для того, чтобы получить общее представление о том, что, по вашему мнению, может сделать с ними поисковая система. Это все равно, что пытаться заглянуть в черный ящик: вы почти ничего не видите, но пытаетесь делать предположения относительно того, что происходит внутри.

В большинстве случаев у пользователей нет возможности понять, что делает поисковая система. Хуже того, все сильно меняется в зависимости от того, как пользователь выражает свой запрос.

Теперь, когда вы понимаете, как обычно работает поисковая система, и вы узнали о некоторых важных проблемах, которые еще не были решены до конца, пришло время познакомиться с глубоким обучением и выяснить, как оно может помочь решить или хотя бы смягчить такие проблемы. Мы начнем с общего обзора возможностей глубоких нейронных сетей.

1.8. Глубокое обучение спешит на помощь

До сих пор мы исследовали темы поиска информации, которые необходимы, чтобы подготовить вас к путешествию по поиску на базе нейронных сетей. Теперь вы приступите к знакомству с глубоким обучением, которое может помочь при создании более интеллектуальных поисковых систем. Этот раздел познакомит вас с основными понятиями глубокого обучения.

В прошлом ключевая проблема *компьютерного зрения* (область компьютерных наук, которая занимается обработкой и пониманием визуальных данных, таких как изображения или видео) при работе с изображениями заключалась в том, что было почти невозможно получить представление изображения, содержащее информацию о закрытых объектах и визуальных структурах. Как заставить компьютер сказать, представляет ли изображение бегущего льва, холодильник, стаю обезь-

ян и т. д.? Глубокое обучение помогло решить эту проблему с помощью создания особого типа глубокой нейронной сети, который мог бы изучать представления изображений постепенно, по одной абстракции за раз, как показано на рис. 1.9.

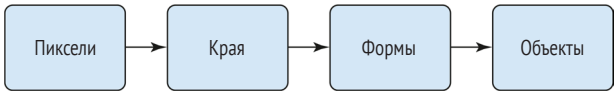


Рис. 1.9 ❖ Постепенное изучение абстракций изображений

Как упоминалось ранее в этой главе, глубокое обучение – это подобласть машинного обучения, которая фокусируется на изучении глубоких представлений текста, изображений или данных в целом путем изучения последовательных абстракций все более значимых представлений. Это делается с помощью глубокой нейронной сети (на рис. 1.10 показана глубокая нейронная сеть с тремя скрытыми слоями). Помните, что нейронная сеть считается *глубокой*, если в ней есть как минимум два скрытых слоя.

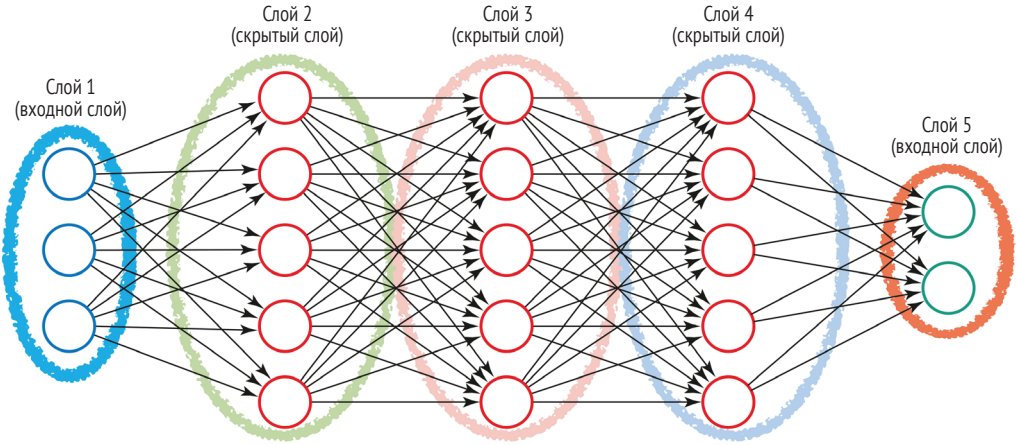


Рис. 1.10 ❖ Глубокая нейронная сеть прямого распространения с тремя скрытыми слоями

На каждом этапе (или слое сети) такие глубокие нейронные сети способны захватывать все более сложные структуры данных. Не случайно компьютерное зрение является одной из областей, которая способствовала разработке и исследованию алгоритмов для изучения представлений при работе с изображениями.

Исследователи обнаружили, что имеет смысл использовать такие глубокие сети, особенно для высококомпозиционных данных. Это означает, что они могут очень помочь, когда вы рассматриваете нечто, что образовано более мелкими частями подобных компонентов. Изображения и текст являются хорошими примерами композиционных данных, потому что их можно постепенно разделить на более мелкие единицы (например, текст → параграфы → предложения → слова). Но (глубокие) нейронные сети не полезны только для изучения представлений; их можно использовать для выполнения множества различных задач в машинном

обучении. Я упомянул, что задача по классификации документов может быть решена с помощью методов машинного обучения.

Несмотря на то что существует множество разных способов создания нейронной сети, нейронные сети обычно состоят из:

- набора нейронов;
- набора связей между всеми или некоторыми нейронами;
- веса (действительное число) для каждой направленной связи между двумя нейронами;
- одной или нескольких функций, которые отображают, как каждый нейрон получает и *распространяет* сигналы по направлению к своим исходящим соединениям;
- при желании набора слоев, которые группируют наборы нейронов, обладающих сходной связностью в нейронной сети.

На рис. 1.10 мы можем идентифицировать 20 нейронов, организованных в набор из 5 слоев. Каждый нейрон в каждом слое связан со всеми нейронами в соседних слоях (предыдущий и последующие слои), за исключением первого и последнего слоев. Традиционно информация начинает течь внутри сети слева направо. Первый слой, который получает входные данные, называется *входным слоем*; и последний слой, называемый *выходным слоем*, выводит результаты нейронной сети. Находящиеся между ними слои называются *скрытыми*.

Представьте, что вы можете применить тот же подход к тексту, чтобы изучить представления документов, которые содержат все более высокие абстракции в документе. Для таких задач существуют методы на базе глубокого обучения, и со временем эти алгоритмы становятся все умнее: их можно использовать для извлечения представлений слов, предложений, абзацев и документов, которые могут охватывать удивительно интересную семантику.

При использовании алгоритма нейронной сети для изучения представлений слов в наборе текстовых документов тесно связанные слова лежат рядом в векторном пространстве. Рассмотрим создание точки на двухмерном графике для каждого слова, содержащегося во фрагменте текста, и посмотрим, как схожие или тесно связанные слова лежат близко друг к другу, как показано на рис. 1.11. Этого можно достичь с помощью алгоритма нейронной сети под названием *word2vec* для изучения таких векторных представлений слов (также называемых *векторами слов*). Обратите внимание, что слова «Информация» и «Поиск» лежат близко друг к другу. Точно так же «word2vec» и «Skip-gram», термы, которые относятся к алгоритмам (неглубокой) нейронной сети, используемым для извлечения векторов слов, находятся рядом друг с другом.

Одной из ключевых идей поиска на основе нейронных сетей является использование таких представлений для повышения эффективности поисковых систем. Было бы неплохо иметь модель поиска, которая опирается на векторы слов и документов (также называющиеся *векторными представлениями*) с этими возможностями, поэтому мы могли бы эффективно рассчитывать и использовать сходства документов и слов, взглянув на *ближайших соседей*. На рис. 1.12 показана глубокая нейронная сеть, используемая для создания представлений слов, содержащихся в проиндексированных документах, которые затем возвращаются в поисковую систему; их можно использовать для настройки порядка результатов поиска.

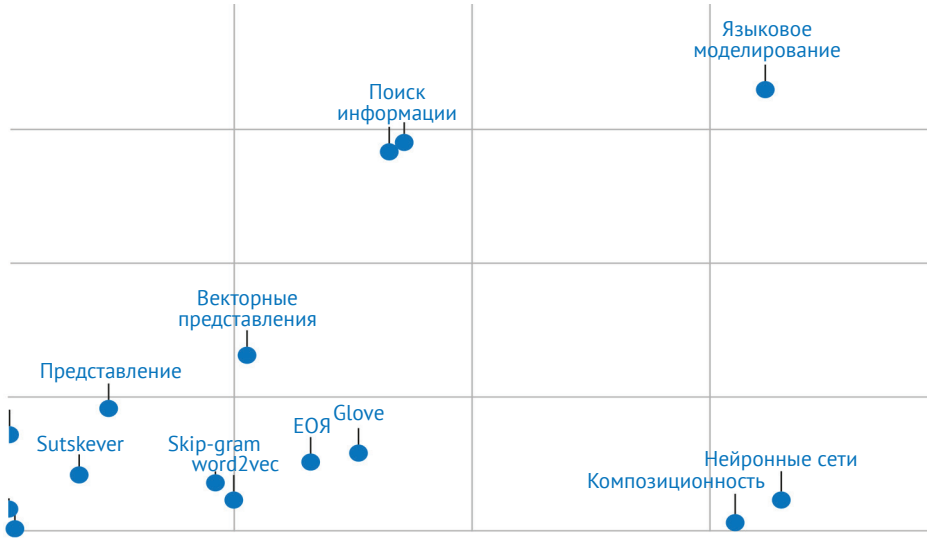


Рис. 1.11 ❖ Векторы слов, полученные из текста исследовательских статей в word2vec

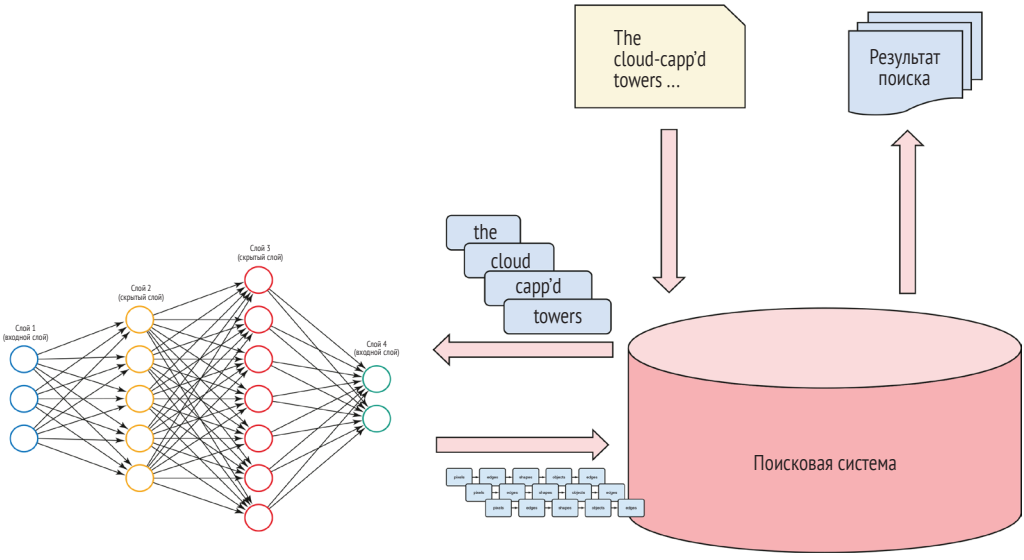


Рис. 1.12 ❖ Приложение для поиска на базе нейронных сетей: использование представлений слов, генерируемых глубокой нейронной сетью, для предоставления более релевантных результатов

В предыдущем разделе анализировалась важность контекста по сравнению со сложностью выражения и понимания информационных потребностей через текстовые запросы. Хорошие семантические представления текста часто строятся с использованием контекста, в котором появляется слово, предложение или

документ, чтобы вывести наиболее подходящее представление. Давайте посмотрим на предыдущий пример, чтобы кратко увидеть, как алгоритмы глубокого обучения могут помочь получить более качественные результаты с релевантностью. Рассмотрим два запроса «последние достижения в области искусственного интеллекта» и «последние достижения в области искусственного интеллекта» из табл. 1.4, предполагая, что мы используем векторную модель. В таких моделях сходство между запросами и документами может сильно варьироваться в зависимости от цепочки анализа текста. Но эта проблема не влияет на векторные представления текста, сгенерированного с помощью современных алгоритмов на базе нейронных сетей. Хотя фраза «искусственный интеллект» и аббревиатура «ИИ» могут находиться далеко друг от друга в векторной модели, они, скорее всего, будут располагаться близко друг к другу, когда построены с использованием представлений слов, генерируемых нейронными сетями. С таким простым изменением мы можем повысить релевантность поисковой системы с помощью более семантически обоснованных представлений слов.

Прежде чем приступить к подробному рассмотрению приложений для поиска на базе нейронных сетей, давайте посмотрим, как поисковые системы и нейронные сети могут работать вместе.

Глубокое обучение и глубокие нейронные сети

Нам нужно провести одно важное различие. Глубокое обучение в основном касается изучения представлений слов, текста, документов и изображений с помощью глубоких нейронных сетей. Глубокие нейронные сети, однако, имеют более широкое распространение: они используются, например, в языковом моделировании, машинном переводе и множестве других задач. В этой книге я буду пояснять, когда мы используем глубокие нейронные сети для изучения представлений и когда мы используем их для других целей. В дополнение к представлениям глубокие нейронные сети могут помочь решить ряд задач, связанных с поиском информации.

1.9. Индекс, пожалуйста, познакомьтесь с нейроном

Искусственная нейронная сеть может научиться прогнозировать результаты на основе обучающего набора с помеченными данными (контролируемое обучение, где каждый ввод снабжен информацией об ожидаемом результате), или она может выполнять обучение без учителя (никакой информации о правильном выводе для каждого ввода не предоставляется), для того чтобы извлекать шаблоны и/или изучать представления. Типичный рабочий процесс поисковой системы включает в себя индексацию и поиск контента; примечательно, что такие задачи могут происходить параллельно. Хотя на данный момент это может показаться технической деталью, то, как вы интегрируете поисковую систему с нейронной сетью, в принципе, важно, поскольку это влияет на эффективность и производительность конструкции поиска. У вас может быть сверхточная система, но если она медлительна, никто не захочет ее использовать! В этой книге вы увидите несколько способов интеграции нейронных сетей и поисковых систем:

- *обучение, затем индексация* – сначала обучите сеть на наборе документов (тексты, изображения), а затем индексируйте те же данные в поисковую

систему и используйте нейронную сеть вместе с поисковой системой во время поиска;

- *индексация, затем обучение* – сначала индексируйте коллекцию документов в поисковой системе; затем обучите нейронную сеть с помощью индексированных данных (в конечном итоге переобучая ее при изменении данных); а потом используйте нейронную сеть вместе с поисковой системой во время поиска;
- *обучение – извлечение – индексация* – сначала обучите сеть на наборе документов и используйте обученную сеть для создания полезных ресурсов, которые будут индексироваться вместе с данными. Поиск происходит как обычно, только с поисковой системой.

Вы увидите, что каждый из этих вариантов в данной книге применяется в правильном контексте. Например, вариант *обучение, затем индексация* будет использоваться в главе 3 для генерации текста, а вариант *индексация, затем обучение* будет использоваться в главе 2 для генерации синонимов из проиндексированных данных. Вариант *обучение – извлечение – индексация* имеет смысл, когда вы используете нейронную сеть для изучения чего-то вроде семантического представления данных, подлежащих индексации; вы будете применять такие представления во время поиска, не прибегая ко взаимодействию с нейронной сетью. Это относится к сценарию, описанному в главе 8 для поиска изображений. В последней главе книги также кратко рассматривается, как справиться с ситуациями, когда поначалу данные доступны не полностью, а поступают в потоковом режиме.

1.10. Обучение нейронной сети

Чтобы использовать мощные возможности обучения нейронной сети, необходимо обучить ее. Обучение сети, подобной той, что показана в предыдущем разделе, происходит с помощью средств контролируемого обучения, предоставляющих входные данные для входного слоя сети, сравнивая (прогнозируемые) результаты сети с известными (целевыми) результатами и позволяя сети учиться на расхождениях между прогнозируемыми и целевыми результатами. Нейронные сети могут легко представлять множество интересных математических функций; это одна из причин, по которой они могут иметь очень высокую верность. Такие математические функции регулируются *весами* соединений и *функциями активации* нейронов. Алгоритм обучения нейронной сети принимает несоответствия между желаемыми и фактическими результатами и корректирует веса каждого слоя, чтобы уменьшить ошибку вывода в будущем. Если вы подадите достаточно данных в сеть, она сможет достичь очень маленького уровня ошибок и, следовательно, хорошо работать. Функции активации влияют на способность нейронной сети выполнять предсказания и скорость обучения; функции активации контролируют, когда и сколько входящего сигнала на нейрон распространяется по всем выходным соединениям.

Наиболее часто используемый алгоритм обучения для нейронных сетей называется методом *обратного распространения ошибки*. При заданных желаемых и фактических результатах алгоритм распространяет *ошибку* каждого нейрона обратно и, следовательно, корректирует свое внутреннее состояние на соединениях

каждого нейрона, по одному слою за раз, от вывода к вводу (назад); *см.* рис. 1.13. Каждый обучающий пример заставляет обратное распространение ошибки «регулировать» состояние каждого нейрона и связи, чтобы уменьшить количество ошибок, создаваемых сетью для этой пары конкретного ввода и желаемого вывода. Это высокоуровневое описание того, как работает алгоритм обратного распространения ошибки; мы подробнее рассмотрим его в последующих главах, когда вы лучше познакомитесь с нейронными сетями.

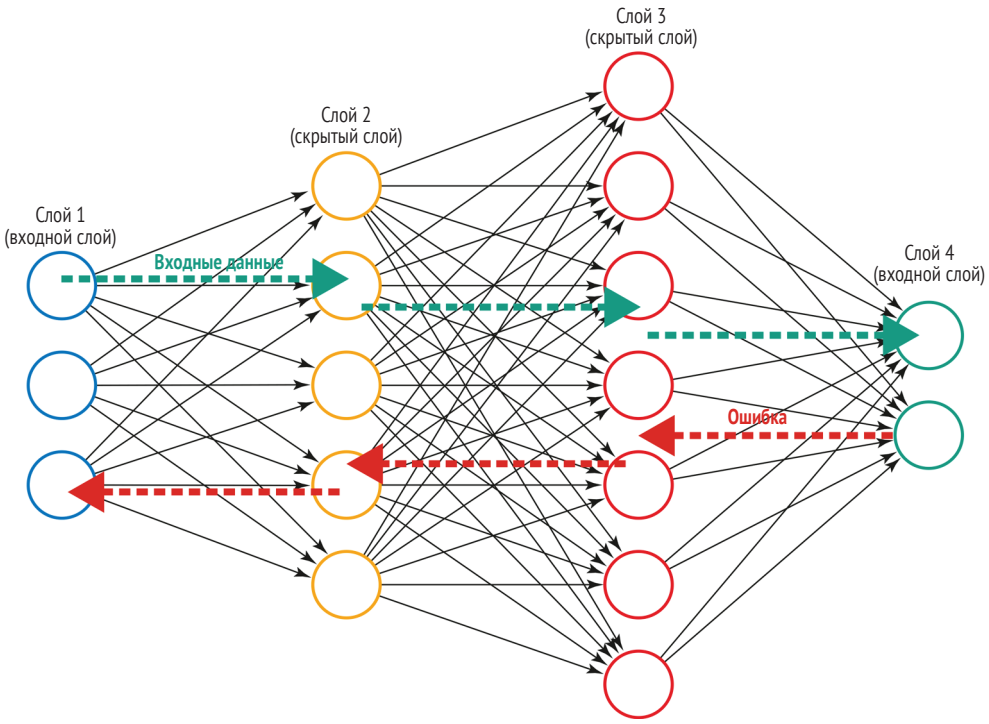


Рис. 1.13 ❖ Шаг вперед (передача входных данных) и шаг назад (обратное распространение ошибки)

Теперь, когда вы понимаете, как обучаются нейронные сети, вам нужно решить, как подключиться к поисковой системе. Поисковые системы могут получать данные для постоянной индексации; поскольку новый контент добавляется, существующий контент обновляется или даже удаляется. Хотя этот процесс относительно легко и быстро поддерживать в поисковой системе, многие алгоритмы машинного обучения создают *статические модели*, которые нельзя быстро адаптировать при изменении данных. Типичный рабочий процесс разработки для задачи машинного обучения включает в себя следующие шаги:

1) выбор и сбор данных для использования в качестве учебного набора;
2) сохранение отдельных частей учебного набора отдельно для оценки и настройки (наборы для тестирования и перекрестной проверки);
3) обучение нескольких моделей машинного обучения в соответствии с алгоритмами (нейронные сети прямого распространения, метод опорных век-

торов и т. д.) и гиперпараметрами (например, количество слоев и количество нейронов в каждом слое для нейронных сетей);
4) оценка и настройка модели для наборов для тестирования и перекрестной проверки;
5) выбор наиболее эффективной модели и использование ее для решения желаемой задачи.

Как видите, этот процесс направлен на создание вычислительной модели, которая будет использоваться для решения определенной задачи или проблемы с использованием статических данных обучения; обновления обучающих наборов (добавленные или измененные входные данные и результаты) таких моделей часто требуют повторения всей последовательности шагов. Это приводит к конфликту с такими системами, как поисковики, которые имеют дело с постоянным потоком новых данных. Например, поисковая система онлайн-газеты будет ежедневно обновляться множеством различных новостей; вы должны принять это во внимание при разработке системы поиска на базе нейронных сетей. Нейронные сети – это модели машинного обучения: вам может потребоваться провести повторное обучение модели или найти решения, чтобы ваша нейронная сеть могла выполнять онлайн-обучение (повторное обучение не требуется)[1].

Рассмотрим, как эволюционировали во времени значения определенных английских слов. Например, слово «cell» сегодня обычно относится к мобильным телефонам или ячейкам с биологической точки зрения; до изобретения мобильных телефонов слово «cell» упоминалось преимущественно по отношению к биологическим клеткам или… тюрьмам! Некоторые концепции тесно связаны со словами только в определенных временных рамках: государственные посты меняются каждые несколько лет, поэтому Барак Обама был президентом Соединенных Штатов в период с 2009 по 2017 год, тогда как термин «президент Соединенных Штатов» применялся по отношению к Джону Фицджеральду Кеннеди между 1961 и 1963 годом. Если подумать о книгах, содержащихся в библиотечном архиве, сколько из них содержат фразу «президент Соединенных Штатов»? Они редко будут относиться к одному и тому же человеку из-за того, что были написаны в разное время.

Я упомянул, что нейронные сети можно использовать для генерации *векторов слов*, которые фиксируют семантику слов, так что слова со сходными значениями будут иметь векторы слов, близкие друг к другу. Как вы думаете, что случится с вектором слов «президент США», если вы обучите модель новостным статьям 1960-х годов и сравните ее с векторами слов, сгенерированными моделью, обученной новостным статьям 2009 года? Будет ли вектор слов «Барак Обама» из последней модели находиться рядом с вектором «президент США» из предыдущей модели? Вероятно, нет, если вы не проинструктируете нейронную сеть, как работать с эволюцией слов во времени[2]. С другой стороны, обычные поисковые

---

[1] *См.*, например: *Andrey Besedin et al.* Evolutive deep models for online learning on data streams with no storage // Workshop on Large-scale Learning from Data Streams in Evolving Environments. 2017 (**http://mng.bz/K14O**) и *Doyen Sahoo et al.* Online Deep Learning: Learning Deep Neural Networks on the Fly (**https://arxiv.org/pdf/1711.03705.pdf**).

[2] *См.*, например: *Zijun Yao et al.* Dynamic Word Embeddings for Evolving Semantic Discovery // International Conference on Web Search and Data Mining. 2018 (**https://arxiv.org/abs/1703.00607**).

системы могут легко работать с такими запросами, как «президент США», и возвращать результаты поиска, содержащие такую фразу, независимо от того, когда они были включены в инвертированный индекс.

1.11. Перспективы поиска на базе нейронных сетей

Поиск на базе нейронных сетей – это интеграция глубокого обучения и глубоких нейронных сетей в поиск на разных этапах. Способность глубокого обучения улавливать глубокую семантику позволяет нам получать релевантные модели и функции ранжирования, которые хорошо адаптируются к базовым данным. Глубокие нейронные сети могут изучить представления изображений, которые дают удивительно хорошие результаты при поиске изображений. Простые меры сходства, такие как косинусное расстояние, могут применяться к представлениям данных, генерируемым с помощью глубокого обучения, для захвата семантически похожих слов, предложений, абзацев и т. д.; здесь есть ряд применений, например на этапе анализа текста и при рекомендации похожих документов. В то же время глубокие нейронные сети могут делать больше, чем «просто» изучать представления; они могут учиться генерировать или переводить текст, а также научиться оптимизировать работу поисковой системы.

Как вы увидите в процессе чтения книги, поисковая система состоит из разных компонентов, которые действуют сообща. Наиболее очевидные части – это ввод данных в поисковую систему и их поиск. Нейронные сети могут использоваться во время индексации для улучшения данных непосредственно перед их поступлением в инвертированный индекс, или их можно использовать для расширения либо указания области действия поискового запроса, чтобы предоставить большее количество результатов или более точные результаты. Но нейронные сети также можно использовать, чтобы давать пользователям умные подсказки, помогая им набирать запросы или переводить свои поисковые запросы «под капотом» и заставить поисковую систему работать с несколькими языками.

Все это звучит потрясающе, но нельзя бросить нейронные сети в поисковую систему и ожидать, что она станет автомагически совершенной. Каждое решение должно приниматься в контексте; а у нейронных сетей есть некоторые ограничения, в том числе стоимость обучения, обновление моделей и многое другое. Но применение поиска на базе нейронных сетей к поисковой системе – это отличный способ сделать ее лучше для пользователей. Это также увлекательное путешествие для специалистов по разработке поисковых систем, которые изучают красоту нейронных сетей.

Резюме

- Поиск – сложная проблема: общие подходы к поиску информации сопряжены с ограничениями и недостатками, и пользователям, и специалистам по разработке поисковых систем может быть трудно заставить все работать так, как ожидалось.
- Анализ текста является важной задачей в поиске как на этапе индексации, так и на этапе поиска, поскольку он подготавливает данные для хранения в ин-

вертированных индексах и оказывает значительное влияние на эффективность поисковой системы.

- Релевантность – это фундаментальная мера того, насколько хорошо поисковая система реагирует на информационные потребности пользователей. Некоторые модели поиска информации могут дать стандартизированную оценку важности результатов для запросов, но универсального средства не существует. Контекст и мнения могут значительно различаться от пользователя к пользователю, поэтому специалисты по разработке поисковых систем должны быть постоянно сосредоточены на измерении релевантности.
- Глубокое обучение – это область машинного обучения, которая использует глубокие нейронные сети для изучения (глубоких) представлений контента (текста, такого как слова, предложения и абзацы, но также и изображений), который может охватывать семантически релевантные меры сходства.
- Поиск на основе нейронных сетей представляет собой мост между поиском и глубокими нейронными сетями с целью использования глубокого обучения, чтобы помочь улучшить различные задачи, связанные с поиском.