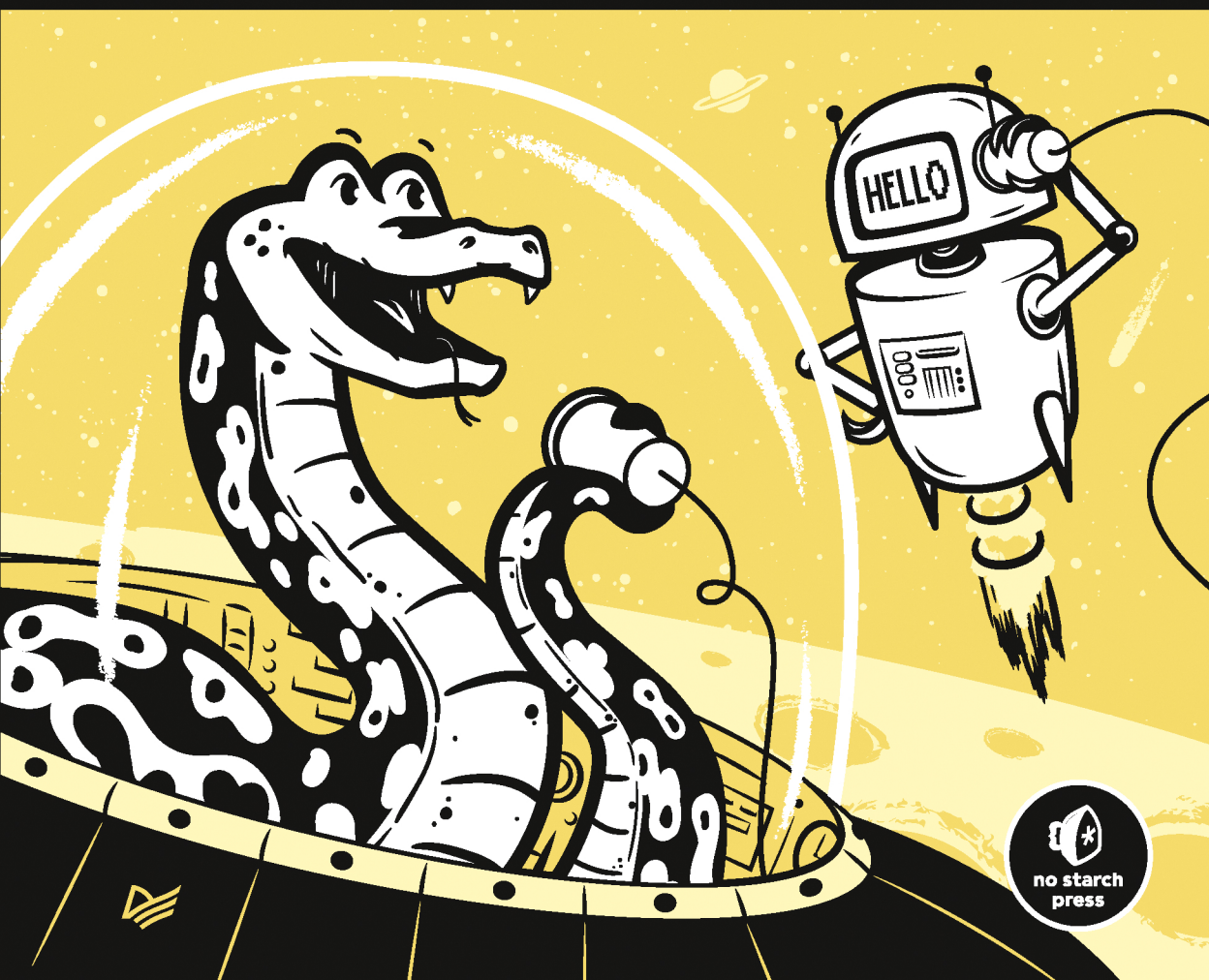
ОБРАБОТКА ЕСТЕСТВЕННОГО ЯЗЫКА PYTHON И SPACY

НА ПРАКТИКЕ

ЮЛИЙ ВАСИЛЬЕВ



ББК 81.111
УДК 81.322.2+004.43
В19

Васильев Юлий

В19 Обработка естественного языка. Python и spaCy на практике. — СПб.: Питер, 2021. — 256 с.: ил. — (Серия «Библиотека программиста»).

ISBN 978-5-4461-1506-8

Python и spaCy помогут вам быстро и легко создавать NLP-приложения: чат-боты, сценарии для сокращения текста или инструменты принятия заказов. Вы научитесь использовать spaCy для интеллектуального анализа текста, определять синтаксические связи между словами, идентифицировать части речи, а также определять категории для имен собственных. Ваши приложения даже смогут поддерживать беседу, создавая собственные вопросы на основе разговора.

Прочитав эту книгу, вы можете сами расширить приведенные в книге сценарии, чтобы обрабатывать разнообразные варианты ввода и создавать приложения профессионального уровня.

16+ (В соответствии с Федеральным законом от 29 декабря 2010 г. № 436-ФЗ.)

ББК 81.111
УДК 81.322.2+004.43

ISBN 978-1718500525 англ.

ISBN 978-5-4461-1506-8

Оглавление

1 Как происходит обработка текстов на естественном языке

В XIX веке на острове Рапа Нуи (более известном как остров Пасхи) исследователи обнаружили деревянные дощечки с письменами *ронго-ронго* — системой загадочных иероглифов. Ученым пока не удалось их расшифровать и определить, что это — письмо или протописьменность (пиктографические символы, несущие информацию, но не связанные с языком). Известно, что создатели надписей воздвигли моаи — гигантские каменные человекоподобные фигуры, которыми знаменит остров Пасхи: однако предназначение моаи до сих пор неясно, о нем можно лишь догадываться.

Если письменность, то есть способ описания людьми различных вещей, непонятна, то, скорее всего, непонятными останутся и прочие аспекты их жизни, в том числе то, что они делали и почему.

Обработка текстов на естественном языке (Natural Language Processing, NLP) — направление искусственного интеллекта, нацеленное на обработку и анализ данных на естественном языке и обучение машин взаимодействию с людьми на естественном языке (языке, сформировавшемся естественным путем на протяжении истории). Через создание алгоритмов машинного обучения, предназначенных для работы с более объемными в сравнении с двумя десятками табличек, найденных на Рапа Нуи, наборами неизвестных данных, исследователи изучают, как люди используют язык. Таким образом появляется возможность добиться гораздо большего, чем просто расшифровка древних надписей.

Сегодня с помощью алгоритмов можно вести научные наблюдения за языками, семантика и грамматические правила которых хорошо известны (в отличие от надписей на ронго-ронго), создавая затем приложения для программного «понимания» высказываний на этих языках. С помощью таких приложений коммерческие компании могут освободить людей от утомительных и однообразных задач. Например, приложение может принимать заказы на доставку еды или отвечать на постоянно повторяющиеся вопросы пользователей в рамках технической поддержки.

Неудивительно, что генерация и понимание естественного языка — наиболее многообещающие, но в то же время сложные задачи NLP. В этой книге для создания обработчика текстов на естественном языке используется язык программирования Python. Задача решается с помощью библиотеки spaCy — ведущей библиотеки Python для обработки естественного языка с открытым исходным кодом. Но для начала вы узнаете, что происходит «за кулисами» в процессе создания средства обработки естественного языка.

Как компьютеры понимают естественный язык

Как можно научить компьютеры — бесчувственные машины — понимать человеческий язык и должным образом реагировать на высказывания? Конечно, машины не способны понимать естественный язык так, как человек. Чтобы компьютер мог производить вычислительные операции над языковыми данными, необходима система преобразования слов естественного языка в числовую форму.

Задание соответствий слов и чисел с помощью вложений слов

Вложение слов (word embedding) — методика задания соответствий слов числам. Словам при вложении слов в соответствие ставятся векторы вещественных чисел, а значение слова распределяется по координатам соответствующего вектора слов. Слова со схожим значением близки в векторном пространстве, поэтому можно определять значение слова по его соседям.

Вот фрагмент подобного отображения:

```
the 0.0897 0.0160 -0.0571 0.0405 -0.0696  ...
and -0.0314 0.0149 -0.0205 0.0557 0.0205  ...
of -0.0063 -0.0253 -0.0338 0.0178 -0.0966 ...
to 0.0495 0.0411 0.0041 0.0309 -0.0044    ...
in -0.0234 -0.0268 -0.0838 0.0386 -0.0321 ...
```

В этом фрагменте словам *the*, *and*, *of*, *to* и *in* соответствуют следующие за ними координаты. Если отобразить эти координаты визуально, то близкие по значению слова окажутся соседями на графике. (Но это не значит, что близкие по значению слова будут сгруппированы и в текстовом представлении так же, как в данном фрагменте. Текстовое представление пространства векторов слов обычно начинается с наиболее распространенных слов, например с *the*, *and* и т. д. Именно таким образом генераторы пространства векторов слов располагают слова.)

ПРИМЕЧАНИЕ

Визуальное представление многомерного векторного пространства можно реализовать в виде двухмерной или трехмерной проекции. Для ее создания используются соответственно первые две или три главные компоненты (координаты) вектора. К этому вопросу мы вернемся в главе 5.

При наличии матрицы соответствий слов числовым векторам над этими векторами можно производить арифметические действия. Среди прочего, можно определять *семантическое подобие* (синонимию) слов, предложений и даже целых документов, а затем использовать эту информацию, чтобы выяснить, например, какой теме посвящен текст.

Математически определение семантического подобия между двумя словами сводится к вычислению косинусного коэффициента между соответствующими векторами, то есть вычислению косинуса угла между ними. И хотя подробное описание вычисления семантического подобия выходит за рамки данной книги, некоторые детали работы с векторами слов описаны в главе 5.

Применение машинного обучения для обработки естественного языка

Вычислить числовые компоненты векторов можно с помощью алгоритмов машинного обучения. *Машинное обучение* (machine learning) — подобласть искусственного интеллекта по созданию компьютерных систем, способных автоматически обучаться на основе передаваемых им данных без необходимости программировать те явным образом. Алгоритмы машинного обучения позволяют предсказывать характеристики новых, не встречавшихся им ранее данных, распознавать изображения и речь, классифицировать фотографии и текстовые документы, автоматизировать управление. Также помогают они и при разработке игр.

Благодаря машинному обучению компьютеры могут решать задачи, которые ранее были им не под силу. Представьте, сколько условных операторов `if...else` пришлось бы написать, чтобы научить машину играть в шахматы, используя традиционный подход — с явным указанием на то, что алгоритм должен делать в той или иной ситуации. Пользователи подобного приложения быстро обнаружили бы слабые места в вашей логике и смогли бы выигрывать проще — пока вы не исправили бы код.

Напротив, приложения, в основе которых лежат алгоритмы машинного обучения, не зависят от заранее описываемой логики, а учатся на своих ошибках. Таким образом, основанная на машинном обучении шахматная программа ищет позиции, встречавшиеся ей в предыдущих партиях, и делает ходы, ведущие к оптимальной позиции. Накопленный опыт она хранит в статистической модели, которую мы рассмотрим в разделе «Что такое статистическая модель в NLP» на с. 31.

Помимо генерации векторов слов, машинное обучение в библиотеке spaCy позволяет выполнить три задачи: *разбор синтаксических зависимостей* (определение взаимосвязи между словами в предложении), *частеречную разметку* (выявление существительных, глаголов и иных частей речи) и *распознавание именованных сущностей* (разбиение имен собственных по категориям — люди, организации, местоположения и т. д.). В следующих главах мы поговорим об этом более подробно.

Жизненный цикл обычной системы машинного обучения включает три этапа: обучение модели, контроль и выполнение предсказаний.

Обучение модели

На первом этапе модель обучается путем подачи на вход алгоритма большого массива данных. Чтобы получить результат, заслуживающий доверия, необходимо обеспечить значительный объем входных данных — намного больший, чем количество надписей на табличках ронго-ронго, например. Что касается NLP, такие платформы, как «Википедия» и Google News, содержат достаточно текста для практически любого алгоритма машинного обучения. Но модель, специально предназначенная для вашего конкретного сценария использования, должна обучаться в том числе на действиях пользователей вашего сайта.

На рис. 1.1 приведена общая картина этапа обучения модели.

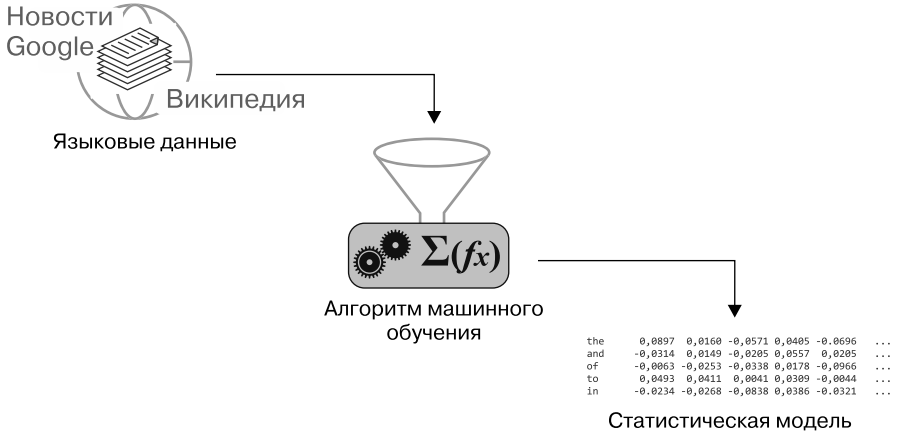


Рис. 1.1. Генерация статистической модели с помощью алгоритма машинного обучения, на основе большого массива текста в качестве входных данных

Чтобы найти слова со схожими характеристиками, модель обрабатывает огромные массивы текстовых данных, а затем создает для этих слов векторы, отражающие общие характеристики.

Из раздела «Что такое статистическая модель в NLP» на с. 31 вы узнаете, что **подобное** пространство векторов слов не единственный компонент статистической модели, предназначенной для NLP. На самом деле ее структура сложнее, что позволяет выделять лингвистические признаки для каждого из слов в зависимости от контекста.

В главе 10 вы узнаете, как обучить и уже существующую (предобученную) модель на новых примерах, и «чистую» модель с самого начала.

Контроль

При желании после обучения модели можно проверить, насколько хорошо она работает. Для контроля работы модели необходимо подать ей на вход текст, который она пока еще не «видела», и проверить, сможет ли она правильно идентифицировать семантические подобия и другие признаки, усвоенные во время обучения.

Выполнение предсказаний

Если все работает как часы, в приложении NLP можно выполнять предсказания на основе обученной модели. Например, предсказать с ее помощью структуру дерева зависимостей для вводимого текста, как показано на рис. 1.2. *Структура дерева зависимостей* отражает взаимосвязи между словами в предложении.

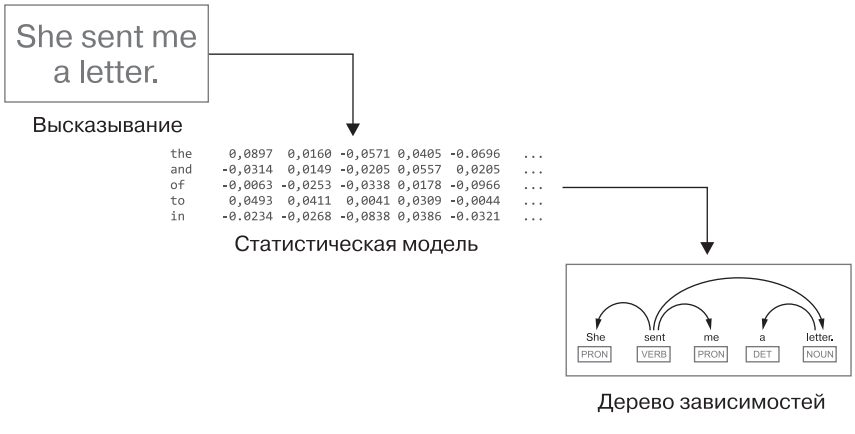


Рис. 1.2. Предсказание структуры дерева зависимостей для высказывания с помощью статистической модели

Наглядно дерево зависимостей можно представить с помощью дуг различной длины, соединяющих синтаксически связанные пары слов. Из приведенного здесь дерева зависимостей видно, что глагол *sent* согласуется с местоимением *she*.

Зачем использовать машинное обучение для обработки естественного языка

Выполняемые алгоритмом предсказания не обычная констатация фактов: предсказания вычисляются с некоторой долей вероятности. Для достижения точности приходится реализовывать все более хитрые алгоритмы, не такие эффективные и менее осуществимые на практике. Обычно стараются достичь разумного баланса безошибочности и быстродействия.

Поскольку идеальное предсказание с помощью моделей машинного обучения недостижимо, может возникнуть вопрос: является ли машинное обучение наилучшим подходом к созданию используемых в приложениях NLP моделей. Другими словами, не существует ли более надежного подхода, в основе которого лежали бы четко заданные правила, аналогичные используемым в компиляторах и интерпретаторах для обработки кода на различных языках программирования? Если коротко: нет. Поясню почему.

Прежде всего, количество слов в языке программирования относительно невелико. Например, в языке Java всего 61 зарезервированное слово, причем смысл каждого из них определен заранее.

Напротив, Оксфордский словарь английского языка, выпущенный в 1989 году, содержит 171 476 словарных статей для используемых в наши дни слов. В 2010 году команда исследователей из Гарвардского университета и Google насчитала около 1 022 000 слов в корпусе оцифрованных текстов, содержащем лишь 4 % когда-либо опубликованных книг. Согласно этому исследованию, словарный состав английского языка ежегодно увеличивается на несколько тысяч слов. Чтобы поставить в соответствие каждому слову число, потребовалось бы очень много времени.

Но даже если попытаться это сделать, существует несколько причин, по которым невозможно определить количество слов естественного языка. Во-первых, непонятно, что именно считать отдельным словом. Например, слово *count* следует считать одним словом, двумя или вообще тремя? В одном контексте оно может обозначать «иметь значение, быть важным», в другом — «произносить числа одно за другим», а в третьем *count* — это существительное.

Следует ли считать различные словоформы — множественное число существительных, времена глаголов и т. д. — отдельными сущностями? Можно ли считать частью языка слова, *заимствованные* из иностранных языков, научные термины, сленг и сокращения? Очевидно, что строгого определения словарного запаса языка не существует, поскольку не так-то просто выяснить, какие группы слов в него нужно включать. В языке программирования наподобие Java попытка включить в код неизвестное слово приведет к ошибке компилятора.

Аналогичная ситуация и с формальными правилами. Как и словарный запас, многие правила любого естественного языка сформулированы достаточно расплывчато и нередко приводят к возникновению неоднозначности. Например, *инфинитивы с отделенной частицей* — грамматическая конструкция, в которой наречие ставится между глаголом в неопределенной форме и его предлогом:

```
spaCy allows you to programmatically extract the meaning of an utterance.
```

В этом примере наречие *programmatically* разделяет предлог и глагол в неопределенной форме *to extract*. Если вы настроены против инфинитивов с отделенной частицей, то можете переписать приведенное предложение следующим образом:

```
spaCy allows you to extract the meaning of an utterance programmatically.
```

Вне зависимости от того, что вы думаете относительно инфинитивов с отделенной частицей, ваше NLP-приложение должно одинаково хорошо понимать оба эти предложения.

Программа же, предназначенная для обработки кода, написанного на каком-либо языке программирования, на решение подобных проблем не рассчитана. Дело в строгой формулировке правил языка программирования, не оставляющей возможности для разночтений. В качестве еще одного примера рассмотрим оператор, написанный на языке программирования SQL, с помощью которого можно вставить данные в таблицу БД:

```
INSERT INTO table1 VALUES(1, 'Maya', 'Silver')
```

Этот оператор говорит сам за себя. Даже если вы не знаете язык SQL, вы легко догадаетесь, что оператор вставляет три значения в таблицу `table1`.

Теперь представьте, что вы изменили его следующим образом:

```
INSERT VALUES(1, 'Maya', 'Silver') INTO table1
```

С точки зрения англоязычного читателя смысл второго оператора не отличается от смысла первого: в конце концов, его можно прочитать как фразу на английском языке и смысл останется прежним. Если же попытаться выполнить второй оператор в SQL-утилите, будет возвращена ошибка `missing INTO keyword` (не найдено ключевое слово `INTO`). Дело в том, что синтаксический анализатор SQL, —как и синтаксический анализатор любого другого языка программирования, —основывается на жестко заданных правилах. Поэтому, чтобы получить желаемое, программист обязан четко описать свои требования. В данном случае синтаксический анализатор SQL ожидает, что ключевое слово `INTO` будет следовать сразу за ключевым словом `INSERT`.

Само собой, в естественном языке подобные ограничения немыслимы. Таким образом, с учетом всех различий совершенно очевидно, что задавать вычислительную модель для естественного языка с помощью набора формальных правил (как это делается с языками программирования) неэффективно или вовсе нереально.

Поэтому наш подход будет основан не на правилах, а на наблюдениях. Вместо кодирования языка путем назначения для каждого слова заранее заданного числа алгоритмы машинного обучения генерируют статистические модели для выявления закономерностей в больших массивах языковых данных и последующего выполнения предсказаний относительно синтаксической структуры новых, еще незнакомых модели текстовых данных.

Рисунок 1.3 резюмирует процессы обработки текстов на естественных языках и языках программирования соответственно.

Система обработки текстов на естественном языке сначала на основе статистической модели выполняет предсказание относительно смысла входного текста, после чего реагирует соответствующим образом. Компилятор же, обрабатывающий код программы, применяет к этому коду набор строго заданных правил.

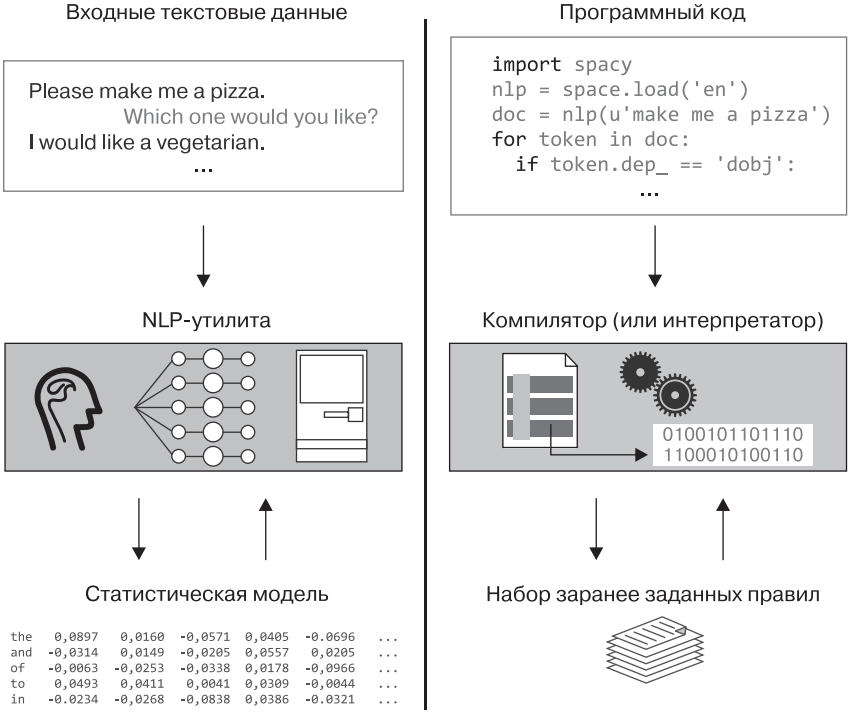


Рис. 1.3. Слева приведен упрощенный технологический процесс обработки текстов на естественном языке, справа — упрощенный технологический процесс обработки кода на языке программирования

Что такое статистическая модель в NLP

В NLP *статистическая модель* (statistical model) содержит оценки распределения вероятностей языковых единиц, например слов или фраз, что позволяет ставить им в соответствие лингвистические признаки. В теории вероятностей и статистике *распределение вероятностей* (probability distribution) для конкретной случайной величины представляет собой таблицу соответствий значений этой величины вероятностям их выпадения (в эксперименте). Таблица 1.1 иллюстрирует пример распределения вероятностей тегов частей речи слова *count* для заданного предложения. (Напомню, что в зависимости от контекста отдельное слово в английском языке может относиться к разным частям речи.)

Таблица 1.1. Пример распределения вероятностей для языковой единицы в конкретном контексте заданного предложения

Глагол	Существительное
78 %	22 %

Конечно, в другом контексте эти числа для слова *count* будут отличаться.

Статистическое моделирование языка играет особо важную роль в решении многих задач обработки естественного языка, в его генерации и понимании. Именно поэтому статистические модели являются краеугольным камнем практически любого NLP-приложения.

На рис. 1.4 в общих чертах показано, как NLP-приложение использует статистическую модель.

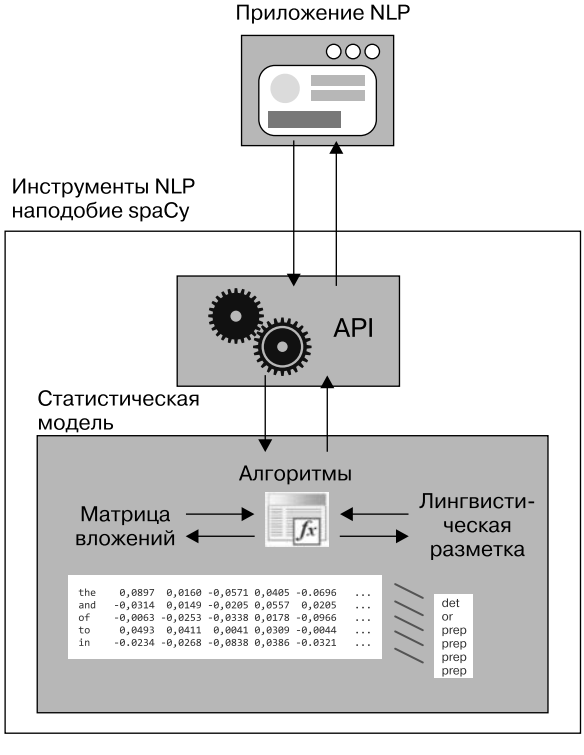


Рис. 1.4. Укрупненный вид архитектуры приложений NLP

Приложение взаимодействует с API spaCy, который абстрагирует статистическую модель, лежащую в его основе. Модель содержит такую информацию, как векторы слов, лингвистическая разметка и т. д. Лингвистическая разметка может иметь различные признаки, например теги частей речи и синтаксическую разметку. В статистической модели также есть набор алгоритмов машинного обучения для извлечения необходимой информации из данных.

На практике данные модели обычно хранятся в двоичном формате. Двоичные данные не рассчитаны на людей, но отлично подходят для машин, так как их можно легко хранить и быстро загружать.

Нейросетевые модели

В таких инструментах NLP, как spaCy, для разбора синтаксических зависимостей, частеречной разметки и распознавания именованных сущностей используются нейросетевые модели. *Нейронная сеть* (neural network) представляет собой набор алгоритмов предсказания. Она состоит из большого числа простых обрабатывающих элементов, подобных нейронам в мозге человека, которые взаимодействуют между собой путем отправки сигналов в соседние узлы и получения встречных сигналов.

Обычно узлы в нейронной сети сгруппированы по слоям: имеются входной и выходной слои, а между ними — один скрытый слой или более. Каждый узел в слое (за исключением выходного слоя) соединяется с каждым узлом из следующего слоя, и каждому соединению соответствует весовой коэффициент. Во время процесса обучения алгоритм подбирает веса таким образом, чтобы минимизировать ошибку предсказаний. Благодаря подобной архитектуре нейронная сеть способна выявлять паттерны даже в сложных входных данных.

По сути, нейронную сеть можно представить так, как на рис. 1.5.

Поступающий сигнал умножается на весовой коэффициент, который представляет собой вещественное число. Источниками передаваемых нейронной сети входных значений и весовых коэффициентов обычно служат векторы слов, сгенерированных во время обучения сети.

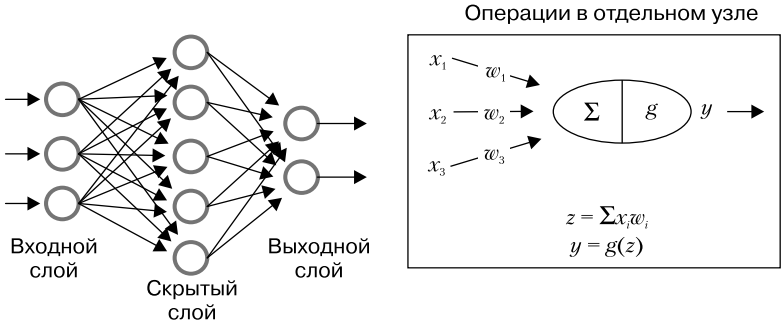


Рис. 1.5. Концептуальная схема нейронной сети и операций, происходящих в отдельном узле

Нейронная сеть складывает результаты этих умножений для всех узлов и передает вычисленную сумму функции активации. Та, в свою очередь, выдает результат (обычно в диапазоне от 0 до 1), генерируя новый сигнал, который затем передается в каждый из узлов последующего слоя, или, в случае выходного слоя, возвращая выходной сигнал. Обычно число узлов выходного слоя равно числу различных возможных исходов для данного алгоритма. Например, количество узлов в нейронной сети, предназначенной для частеречной разметки, должно совпадать с числом поддерживаемых системой тегов частей речи, как показано на рис. 1.6.

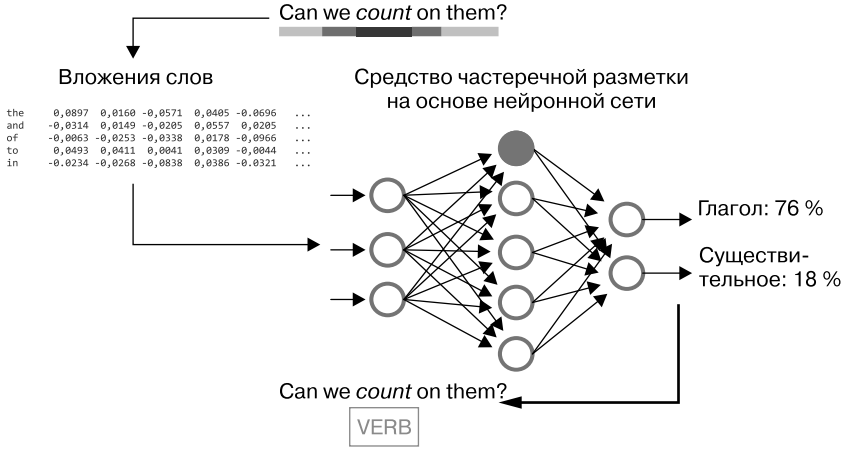


Рис. 1.6. Упрощенная схема процесса частеречной разметки

На выходе частеречной разметки выдается распределение вероятностей по всем возможным частям речи для заданного слова в конкретном контексте.

Использование сверточных нейронных сетей для NLP

Архитектура настоящей нейросетевой модели может быть очень сложной и состоять из большого числа различных слоев. Например, нейросетевая модель, применяемая в spaCy, представляет собой так называемую *сверточную нейронную сеть* (convolutional neural network, CNN), имеющую сверточный слой, который используется и при частеречной разметке, и в синтаксическом анализаторе, и при распознавании именованных сущностей. Сверточный слой применяет к отдельным областям входных данных набор фильтров обнаружения, проверяя, присутствуют ли в этих данных определенные признаки.

Посмотрим, как могла бы работать CNN при задаче частеречной разметки предложения из предыдущего примера:

```
Can we count on them?
```

Вместо того чтобы анализировать каждое слово по отдельности, сверточный слой сначала разбивает предложение на части. Предложение в NLP можно считать матрицей, каждая строка которой соответствует слову, представленному в виде вектора. Таким образом, если размерность каждого из векторов слов равна 300, а длина предложения — пять слов, получится матрица размером 5×300. Если размер фильтра обнаружения в сверточном слое равен 3 (то есть он применяется к трем последовательным словам), размер покрывающих входные данные областей составит 3×300. Такого контекста будет достаточно, чтобы соотнести каждое слово с тегом части речи.

Операция частеречной разметки с помощью сверточного подхода показана на рис. 1.7.

В предыдущем примере самым сложным для средства разметки было определить, к какой части речи относится слово *count*. Проблема в том, что в зависимости от контекста это слово может быть и глаголом, и существительным. Но задача резко упрощается, когда процедура

разметки видит фрагмент с сочетанием слов *we count on*, из которого ясно, что *count* может быть только глаголом.

Рис. 1.7. Упрощенная схема применения сверточного подхода к задаче NLP

Подробный рассказ обо всем происходящем «под капотом» сверточной архитектуры выходит за рамки данной книги. Узнать больше об архитектуре нейросетевых статистических моделей spaCy можно из раздела Neural Network Model Architecture («Архитектура нейросетевых моделей») документации API spaCy.

Какие задачи остаются за вами?

В предыдущем разделе вы узнали, что библиотека spaCy использует нейросетевые модели для разбора синтаксических зависимостей, для частеречной разметки и для распознавания именованных сущностей. Поскольку spaCy решает эти задачи сама, что же остается вам как разработчику NLP-приложения?

Одна из вещей, которую библиотека spaCy не может сделать за вас, — распознать намерения пользователя. Допустим, вы занимаетесь продажей одежды и онлайн-приложение для приема заказов получило следующий запрос от клиента:

```
I want to order a pair of jeans.
```

Ваше приложение должно распознать, что клиент намерен разместить заказ на пару джинсов.

Если воспользоваться spaCy для разбора синтаксических зависимостей приведенной фразы, получится результат, показанный на рис. 1.8.

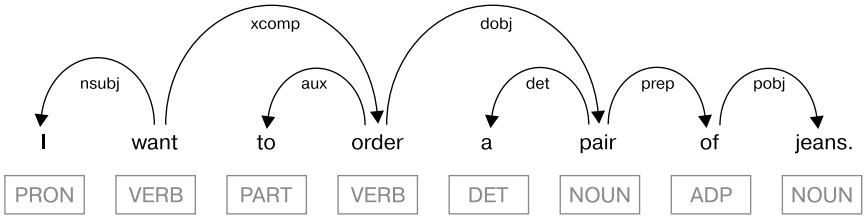


Рис. 1.8. Дерево зависимостей для нашего примера высказывания

Обратите внимание, что spaCy не отмечает элементы сгенерированного дерева как отражающие намерения пользователя. Иное было бы даже странно. Дело в том, что spaCy не знает, как именно вы реализовали логику приложения и какую разновидность намерений хотели бы видеть. Определять ключевые слова для решения задачи по распознаванию намерений можете только вы.

Для распознавания смысла высказывания или текста важны следующие основные аспекты: ключевые слова, контекст и переход значения.

Ключевые слова

Чтобы выбрать наиболее важные слова для распознавания смысла, можно воспользоваться результатами разбора синтаксических зависимостей. В примере с фразой *I want to order a pair of jeans* ключевыми словами, очевидно, являются *order* и *jeans*.

Обычно для формирования намерения вполне достаточно переходного глагола и его прямого дополнения. Но в данном примере ситуация более запутанная. Необходимо пройтись по дереву зависимостей и извлечь *order* (переходный глагол) и *jeans* (предложное дополнение, относящееся к прямому дополнению *pair*).

Контекст

Контекст может играть важную роль при выборе ключевых слов, поскольку смысл одной и той же фразы может быть разным в разных контекстах. Допустим, нужно обработать следующее высказывание:

```
I want the newspaper delivered to my door.
```