ГЛАВА 24

Глубокое обучение при обработке естественного языка

В этой главе объясняется, как глубокие нейронные сети можно использовать для решения различных задач обработки языка, благодаря их способности к захвату и фиксации структуры естественного языка наряду с его гибкостью и текучестью.

В главе 23 рассматривались ключевые элементы естественного языка, в том числе грамматика и семантика. Системы, основанные на синтаксическом и семантическом анализе, продемонстрировали успехи при решении многих задач, но их производительность ограничена бесконечной сложностью лингвистических феноменов в реальных текстах. Учитывая огромное количество текста, доступного в форме, подходящей для машинного прочтения, имеет смысл проанализировать, могут ли подходы, основанные на машинном обучении, управляемом данными, быть более эффективными. Исследование этого предположения будет проведено с использованием инструментов, предоставляемых системами глубокого обучения (глава 21).

В разделе 24.1 показано, как можно улучшить обучение, представив слова не как атомарные значения, а как точки в многомерном пространстве. В разделе 24.2 обсуждается использование рекуррентных нейронных сетей для захвата смыслового значения и дальнего контекста на больших расстояниях в тех случаях, когда текст обрабатывается последовательно. Раздел 24.3 посвящен в основном машинному переводу, одной из наиболее успешных областей применения глубокого обучения к задачам обработки естественного языка. В разделах 24.4 и 24.5 рассматриваются такие модели, которые можно обучить на основании большого количества неразмеченного текста, а затем успешно применить к конкретным задачам с достижением в конечном итоге современного уровня производительности. Наконец, в разделе 24.6 подводится итог тому, какие рубежи в области обработки естественного языка уже достигнуты и в каком направлении может идти дальнейшее развитие.

24.1. Метод встраивания слов

Необходимо найти такое представление слов, которое не требует ручной разработки признаков, но позволяет обобщать связанные слова, т.е. слова, которые связаны синтаксически (“colorless” (*бесцветный*) и “ideal” (*идеальный*) — прилагательные), семантически (“cat” (*кошка*) и “kitten” (*котенок*) относятся к кошачьим), тематически (“sunny” (*солнечный*) и “sleet” (*дождь со снегом*) определяют погодные условия), с точки зрения выражения мнения (“awesome” (*здорово*) противоположно “miserably” (*ужасно*)) или иным образом.

Как следует закодировать слово во входном векторе **x**, чтобы его можно было обрабатывать в нейронной сети? В разделе 21.2.1 для этой цели предлагалось использовать **вектор быстрого** (однократного) **кодирования** (*one-hot vector*), т.е. кодировать *i*-е слово в словаре единичным значением бита в *i*-й позиции входного вектора с нулевыми значениями во всех остальных его битах. Но такое представление не позволяет выявлять сходства между словами.

Следуя изречению лингвиста Джона Р. Ферта ([744], 1957) “Вы узнаете слово по его окружению”, можно было бы представить каждое слово вектором *n*-граммных счетчиков всех фраз, в которых это слово появляется. Однако необработанные *n*-граммные счетчики крайне громоздки. Для словаря со 100 тыс. слов потребуется 10^{25} 5-грамм для отслеживания (хотя векторы в этом 10^{25}-мерном пространстве были бы довольно разреженными — большинство счетчиков были бы нулевыми). Можно получить лучшее обобщение, если уменьшить это представление до вектора меньшего размера, возможно, лишь с несколькими сотнями измерений. Такое меньшее, но более плотное представление получило название “вектор ►**встраивания слов**” (*word embedding*). Встраивания слов *изучаются автоматически* на основании данных (позже будет показано, как это делается.) Что представляют собой эти изученные встраивания слов? С одной стороны, каждое из них — просто вектор чисел, в котором отдельные измерения и их числовые значения не имеют какого-либо различимого смысла:

“aardvark” = [–0,7; +0,2; –3.2;…]
“abacus” = [+0,5; +0,9; –1,3;…]
…
“zyzzyva ” = [–0,1; +0,8; –0,4;…].

С другой стороны, пространство признаков обладает тем свойством, что в конечном итоге схожие слова будут иметь сходные векторы. Это можно увидеть на рис. 24.1, на котором легко различаются отдельные кластеры для названий стран (france, greece, germany), отношений родства (nephew, niece, uncle, aunt), транспорта (car, bicycle, truck) и еды (banana, pizza, apple).

Как оказалось, по причинам, которые остаются не вполне понятными, у векторов встраивания слов есть и дополнительные свойства, помимо простой близости подобных слов. Предположим, например, что рассматриваются векторы **A** для

Афин и **B** для Греции. Складывается впечатление, что для этих слов разность векторов **B** – **A** кодирует отношение “страна/столица”. Для других подобных пар — Франция и Париж, Россия и Москва, Замбия и Лусака — разность соответствующих векторов, по существу, одна и та же.

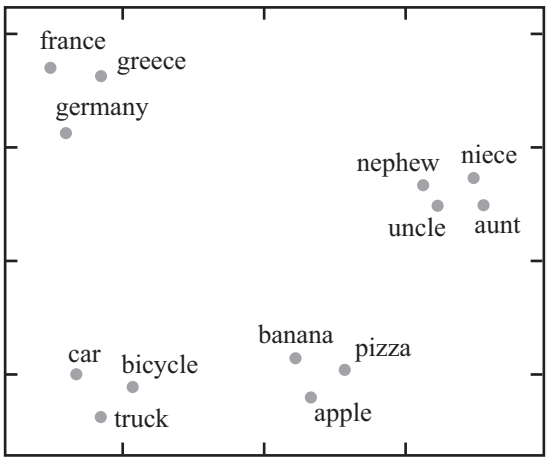


Рис. 24.1. Векторы встраивания слов, вычисленные по алгоритму GloVe (обучение выполнялось на корпусе текста с 6 млрд слов). В этой визуализации 100-мерные векторы слов спроецированы на плоскость с двумя измерениями, — подобные по смыслу слова располагаются рядом друг с другом

Это свойство можно использовать для решения задач поиска аналогичных слов, таких как “Афины для Греции как Осло для [*что*]?” Обозначим вектор слова “Осло” как **C** и вектор искомого слова как **D**, тогда можно предположить, что **B**–**A** = **D** – **C**, что дает **D** = **C** + (**B** – **A**). Если вычислить этот новый вектор **D**, то можно обнаружить, что его значение оказалось ближе к вектору слова “Норвегия”, чем к любому другому вектору. На рис. 24.2 показано, что такой тип векторной арифметики работает для многих отношений.

Тем не менее нет никакой гарантии, что конкретный алгоритм встраивания слов, работающий с заданным корпусом текста, зафиксирует определенные семантические отношения. Метод встраивания слов популярен потому, что доказал свою способность давать результат, являющийся хорошим представлением для последующих языковых задач (таких, как предоставление ответов на вопрос, перевод или обобщение), а не потому, что он сам по себе гарантированно обеспечивает ответы на аналогичные вопросы.

Использование векторов встраивания слов вместо их однократного кодирования оказалось полезным практически для всех приложений глубокого обучения, направленных на решение задач обработки естественного языка. Действительно, во

многих случаях оказалось возможным использовать общие **предварительно обученные** векторы, полученные от любого из нескольких возможных поставщиков, для решения собственных конкретных задач обработки языка. На момент написания этой книги к чаще всего используемым предобученным векторным словарям можно отнести WORD2VEC, GloVe (Global Vectors) и FASTTEXT, включающий встраивания для 157 языков. Использование предобученной модели позволяет сэкономить много времени и усилий. Больше информации об этих ресурсах можно найти в разделе 24.5.1.

A	**B**	**C**	**D** = **C** + (**B** − **A**)	Отношение
Athens	Greece	Oslo	Norway	*Столица*
Astana	Kazakhstan	Harare	Zimbabwe	*Столица*
Angola	kwanza	Iran	rial	*Валюта*
copper	Cu	gold	Au	*Обозначение элемента*
Microsoft	Windows	Google	Android	*Операционная система*
New York	New York Times	Baltimore	Baltimore Sun	*Городская газета*
Berlusconi	Silvio	Obama	Barack	*Фамилия*
Switzerland	Swiss	Cambodia	Cambodian	*Национальность*
Einstein	scientist	Picasso	painter	*Профессия*
brother	sister	grandson	granddaughter	*Отношение родства*
Chicago	Illinois	Stockton	California	*Штат*
possibly	impossibly	ethical	unethical	*Отрицание*
mouse	mice	dollar	dollars	*Множественное число*
easy	easiest	lucky	luckiest	*Превосходная степень*
walking	walked	swimming	swam	*Прошедшее время*

Рис. 24.2. Модель встраивания слов иногда может отвечать на вопрос “**A** для **B** как **C** для [*что*]?” с использованием векторной арифметики: при заданных векторах встраивания для слов **A**, **B** и **C** вычислите вектор **D** = **C** + (**B** − **A**) и найдите слово, наиболее близкое к **D**. (Ответы в столбце **D** были автоматически рассчитаны моделью, а описания в столбце “Отношение” добавлены вручную.) Адаптировано из статей Миколова и соавт. [1568] (2013) и [1570] (2014)

Также можно обучить и собственные векторы встраивания слов, — обычно это делается одновременно с обучением сети для конкретной задачи. В отличие от общих предварительно обученных векторов, встраивания слов, предназначенные для конкретной задачи, могут быть обучены на тщательно отобранном корпусе и в результате приобрести тенденцию подчеркивать те аспекты слов, которые полезны именно для этой задачи. Например, предположим, что речь идет о задаче проставления меток частей речи (POS; см. раздел 23.1.6). Напомним, что ее решение предполагает правильное прогнозирование части речи для каждого слова в предложении. Хотя это довольно простая задача, она вовсе не тривиальна, поскольку в английском языке многие слова могут быть помечены разным способом, например

слово *cut* в предложении может быть глаголом настоящего времени (переходным или непереходным), глаголом прошедшего времени, частью инфинитива, причастием прошедшего времени, прилагательным или существительным. Если соседнее временно*е* наречие относится к прошлому, то можно предположить, что это конкретное вхождение слова *cut* является глаголом прошедшего времени; и тогда можно надеяться, что применение метода встраивания позволит зафиксировать этот аспект наречий — определять отношение к прошлому.

Задача проставления POS-тегов может служить хорошим введением в применение глубокого обучения при обработке естественных языков, поскольку лишена запутанности, характерной для более сложных задач, таких как предоставление ответов на вопросы (см. раздел 24.5.3). При наличии корпуса предложений с проставленными POS-тегами изучение параметров для создания векторов встраивания слов и проставления POS-тегов выполняется одновременно. Этот процесс выполняется следующим образом.

1. Выберите ширину w (нечетное количество слов) для окна прогнозирования, которое будет использоваться для пометки каждого слова. Типичное значение $w = 5$; это означает, что тег для слова прогнозируется на основе двух слов слева и двух слов справа от него. Все предложения в используемом корпусе разделите на перекрывающиеся окна длиной w. Каждое такое окно будет представлять собой один обучающий пример, состоящий из w слов в качестве входных данных и категории POS среднего слова в качестве выхода.
2. Создайте словарь из всех уникальных токенов слов, встречающихся в обучающих данных более пяти раз. Обозначим общее количество слов в словаре как v.
3. Отсортируйте полученный словарь в произвольном порядке (например, в алфавитном).
4. Выберите значение d в качестве размера каждого вектора встраивания слова.
5. Создайте новую весовую матрицу $\mathbf{E}$ размером $v \times d$; это будет матрица встраивания слов. Строка i матрицы $\mathbf{E}$ — это вектор встраивания i-го слова в словаре. Инициализируйте матрицу $\mathbf{E}$ случайными значениями (или воспользуйтесь данными из предварительно обученных векторов).
6. Настройте нейронную сеть, которая выводит метку части речи, как показано на рис. 24.3. Первый слой будет состоять из w копий матрицы встраивания. Далее можно использовать два дополнительных скрытых слоя, $\mathbf{z}_1$ и $\mathbf{z}_2$ (с весовыми матрицами $\mathbf{W}_1$ и $\mathbf{W}_2$ соответственно), за которыми следует слой с функцией активации softmax, дающий выходное распределение вероятности $\hat{\mathbf{y}}$ по возможной категории части речи для среднего слова:

$$\begin{aligned} \mathbf{z}_1 &= \sigma(\mathbf{W}_1\mathbf{x}) \\ \mathbf{z}_2 &= \sigma(\mathbf{W}_2\,\mathbf{z}_1) \\ \hat{\mathbf{y}} &= \text{softmax}\,(\mathbf{W}_{out}\,\mathbf{z}_2). \end{aligned}$$

7. Чтобы закодировать последовательность из *w* слов в качестве входного вектора, просто найдите вектор встраивания для каждого ее слова и конкатенируйте эти векторы. Результатом будет вектор **x** длиной *wd* со значениями, представленными действительными числами. Даже если данное слово будет иметь такой же вектор встраивания, как тот, который находится на первой позиции, последней позиции или где-то посередине, каждый вектор встраивания будет умножен на другую часть первого скрытого слоя; следовательно, имеет место неявное кодирование относительной позиции каждого слова.
8. Обучайте матрицу весов **E** и остальные матрицы весов $\mathbf{W}_1$, $\mathbf{W}_2$ и $\mathbf{W}_{out}$, используя метод градиентного спуска. Если все пойдет хорошо, среднее слово, *cut*, будет помечено как глагол прошедшего времени, исходя из свидетельства в окне *w*, включающего временну*ю* ссылку на прошедшее время в виде слова “yesterday” (*вчера*), субъектное местоимение третьего лица “they” (*они*) непосредственно перед словом *cut*, и т.д.

Альтернативой методу встраивания слов является **модель на уровне символов**, в которой ввод представляет собой последовательность символов, каждый из которых закодирован как вектор быстрого кодирования. Такая модель должна изучать, как символы объединяются с целью формирования слов. Авторы большинства исследовательских работ в области обработки естественных языков НЛП придерживаются уровня слов, а не кодировки на уровне символа.

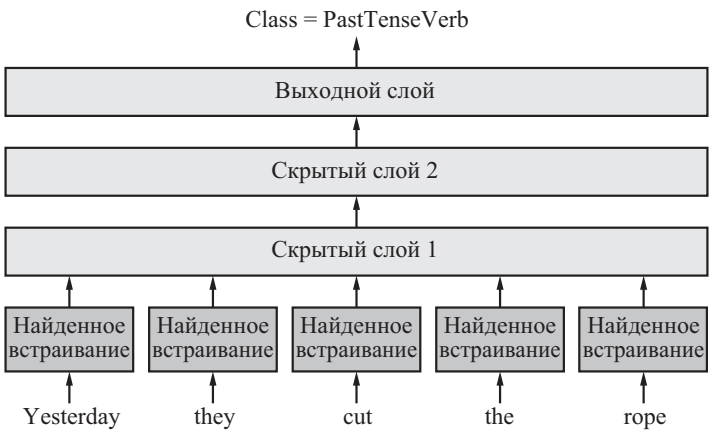


Рис. 24.3. Модель с прямой связью для простановки тегов части речи. В этой модели в качестве входных данных используется окно из пяти слов, а результатом является прогноз тега части речи для слова в середине окна, — в данном случае это *cut*. Модель способна учитывать положение слова, поскольку каждое из пяти входных встраиваний умножается на разные части первого скрытого слоя. В процессе обучения значения параметров для создания векторов встраивания слов и для трех слоев сети простановки тегов изучаются одновременно

24.2. Рекуррентные нейронные сети для задач обработки естественного языка

Хотя теперь у нас уже есть хорошее представление для каждого слова в отдельности, язык образуется из упорядоченных последовательностей слов, в которых важен **контекст**, т.е. набор окружающих слов. Для простых задач, таких как пометка слов в предложении тегами части речи, маленькое окно фиксированного размера — например, из пяти слов — обычно предоставляет достаточно контекста для ее решения.

Более сложные задачи, такие как предоставление ответов на вопросы или разрешение ссылок, в качестве контекста могут потребовать десятков слов. Например, в предложении "Eduardo told me that Miguel was very sick so I took **him** to the hospital" (*Эдуардо сказал мне, что Мигель был очень болен, поэтому я отвез* его *в больницу*) выяснение, что слово **him** (*его*) относится к Мигелю, а не к Эдуардо, требует знания контекста, охватывающего все предложение из 14 слов, от первого до последнего.

24.2.1. Модели языка с рекуррентными нейронными сетями

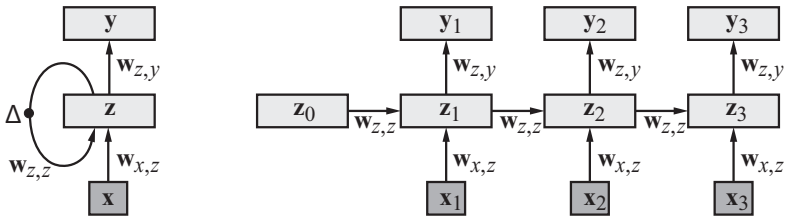
Начнем с проблемы создания **модели языка** с достаточным контекстом. Напомним, что модель языка — это распределение вероятностей по последовательностям слов. Такое решение позволяет прогнозировать следующее слово в тексте, учитывая все предыдущие слова, и часто используется в качестве строительного блока при решении более сложных задач.

Построение модели языка с помощью *n*-граммной модели (см. раздел 23.1) или сети с прямой связью с фиксированным окном из *n* слов может столкнуться с большими затруднениями из-за проблемы контекста: либо требуемый контекст будет превышать фиксированный размер окна, либо модель будет иметь слишком много параметров, либо и то, и другое одновременно.

Кроме того, для сети с прямой связью актуальна проблема **асимметрии**: что бы она ни выучила о, скажем, появлении слова *him* в качестве 12-го слова в предложении, ей придется переучиваться при появлении этого слова в других позициях в предложении, поскольку для каждой позиции веса будут разными.

В разделе 21.6 была представлена **рекуррентная нейронная сеть**, или **RNN**, предназначенная для последовательной обработки данных временны*х* рядов, по одному этапу за раз. Это позволяет предположить, что RNN могут оказаться полезными при обработке естественных языков, по одному слову на каждом этапе. Такая сеть была приведена на рис. 21.8, — здесь она приводится еще раз как рис. 24.4.

В рекуррентной нейронной сети модели языка каждое входное слово кодируется как вектор встраивания слова $\mathbf{x}_t$. В сети имеется скрытый слой $\mathbf{z}_t$, который передает в качестве входных данных информацию от одного временно*го* этапа к следующему. Общая цель заключается в выполнении мультиклассовой классификации, где



а) б)

Рис. 24.4. а) Принципиальная схема базовой рекуррентной нейронной сети, в которой скрытый слой **z** включает рекуррентные соединения, — символ Δ здесь указывает на задержку. **б)** Та же самая сеть, развернутая по трем временны*м* этапам для создания представления в виде сети с прямой связью. Обратите внимание, что одни и те же веса используются на всех временны*х* этапах

классы — слова из словаря. Следовательно, выход $\mathbf{y}_t$ будет распределением вероятности softmax по возможным значениям следующего слова в предложении.

Используя архитектуру RNN, можно решить проблему слишком большого количества параметров. Количество параметров в весовых матрицах $w_{z,z}$, $w_{x,z}$ и $w_{z,y}$ остается постоянным независимо от количества слов, — это $O(1)$. Это важное отличие от сетей с прямой связью, в которых количество параметров определяется как $O(n)$, и n-граммных моделей, в которых количество параметров определяется как $O(v^n)$, где v — размер словаря.

Архитектура RNN также позволяет решить проблему асимметрии, так как веса остаются прежними для каждой позиции слова.

В некоторых случаях архитектура RNN также позволяет решить проблему ограниченного контекста. Теоретически нет предела тому, насколько далеко назад во входных данных может заглядывать модель. Каждое обновление скрытого слоя $\mathbf{z}_t$ имеет доступ как к текущему входному слову $\mathbf{x}_t$, так и к предыдущему состоянию скрытого слоя $\mathbf{z}_{t-1}$, а это означает, что информация о любом слове на входе может храниться в скрытом слое неопределенно долго, копируясь (или соответствующим образом изменяясь) от одного временно*го* шага к другому. Но, конечно, объем памяти в скрытом слое **z** ограничен, поэтому он не может помнить все обо всех предыдущих словах.

На практике модели RNN хорошо справляются с различными задачами, но не со всеми. Может оказаться затруднительным предсказать, будет ли их применение успешным для определенной проблемы. Одним из факторов, способствующих успеху, является то, что учебный процесс должен побуждать сеть выделять пространство для хранения в скрытом слое **z** таких аспектов входных данных, которые действительно окажутся полезными для решения поставленной задачи.

Для обучения модели языка на базе RNN используется процесс обучения, описанный ранее, в разделе 21.6.1. Входными данными $\mathbf{x}_t$ здесь являются слова из обучающего корпуса текста, а наблюдаемыми выходными данными — слова из него же, но смещенные на 1. Иначе говоря, для обучающего текста "hello world" первым входом $\mathbf{x}_1$ является вектор встраивания слова "hello", а первым выходом $\mathbf{y}_1$ будет вектор встраивания для слова "world". Модель обучается предсказывать следующее слово, и ожидается, что для достижения этой цели она будет использовать свой скрытый слой, размещая в нем некую полезную информацию. Как объясняется в разделе 21.6.1, вычисляется разность между наблюдаемым выходом и фактическим выходом, вычисленным сетью, которая затем распространяется обратно во времени, с целью обеспечения сохранения весов одинаковыми для всех временны*ы*х этапов.

После обучения модель можно будет использовать для генерации произвольного текста. Модели передается начальное входное слово $\mathbf{x}_1$, на основании которого она производит выходное слово $\mathbf{y}_1$, представленное как распределение вероятностей softmax по словам. Далее из этого распределения выбирается одно слово, которое записывается в качестве выходного для времени t, а затем передается сети в качестве следующего входного слова $\mathbf{x}_2$. Этот цикл повторяется столько раз, сколько требуется. При выборе слова из распределения $\mathbf{y}_1$ доступно множество вариантов: можно всегда извлекать наиболее вероятное слово, можно выбирать каждое слово в соответствии с его вероятностью или можно иногда выбирать менее вероятные слова, чтобы добавить больше разнообразия в сгенерированный результат. Правило формирования выборки — это гиперпараметр модели.

Вот пример случайного текста, сгенерированного языковой моделью RNN, обученной на текстах Шекспира (Карпати [1190], 2015).

Marry, and will, my lord, to weep in such a one were prettiest;
Yet now I was adopted heir
Of the world's lamentable day,
To watch the next way with his father with his face?

(*Женись, и будешь, милорд, плакать в таком, были самые красивые;*
Все же теперь я был усыновленным наследником
Из мира печального дня,
Чтобы смотреть следующий путь с его отцом и с его лицом?)

24.2.2. Классификация с использованием рекуррентных нейронных сетей

Рекуррентные нейронные сети также можно использовать для решения других задач обработки языка, таких как проставление тегов части речи (POS) или разрешение кореферентности (*coreference resolution*). В обоих случаях входной и скрытый слои будут одинаковыми, но для задачи присвоения тегов части речи на выходе сети будет распределение softmax по тегам POS, тогда как для задач разрешения

кореферентности это будет распределение softmax по возможным предшественникам. Например, если на вход сети поступит слово “его” в предложении “Эдуардо сказал мне, что Мигель был очень болен, поэтому я взял **его** в больницу”, на выходе сети высокая вероятность кореферентности с этим словом должна быть присвоена слову-предшественнику “Мигель”.

Обучение сети RNN для выполнения классификации проводится так же, как и при изучении модели языка. Разница лишь в том, что для обучающих данных дополнительно потребуются метки — метки частей речи или указания на кореферентность. Это значительно затрудняет сбор обучающих данных в сравнении с задачей изучения модели языка, когда все, что требуется, — это непомеченный текст.

Цель обучения модели языка — дать прогноз для *n*-го слова на основании предыдущих слов. Но для задач классификации нет никаких причин, по которым следует ограничиваться рассмотрением только предыдущих слов. Может оказаться очень полезным заглянуть в предложении вперед. В приведенном выше примере кореферентности слово, кореферентное с “его”, было бы совсем иным, если бы предложение заканчивалось словами “навестить Мигеля“, а не “в больницу”, поэтому в таких задачах крайне важно заглядывать вперед. Из экспериментов по отслеживанию движения глаз известно, что читающий человек не следует взглядом по строкам строго слева направо.

Для захвата контекста справа можно использовать ►**двунаправленную RNN** — рекуррентную нейронную сеть, объединяющую посредством конкатенации две отдельные модели: модель, работающую справа налево, и модель, работающую слева направо. Пример использования двунаправленной RNN в задаче расстановки тегов POS приведен на рис. 24.5.

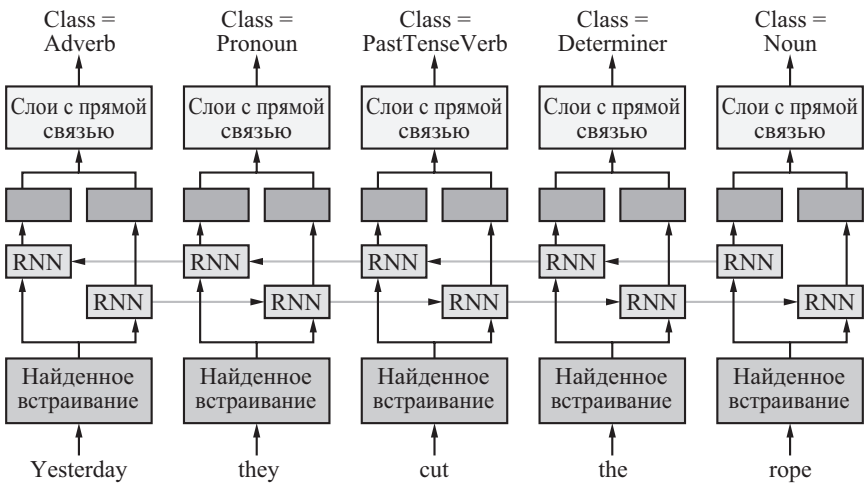


Рис. 24.5. Двунаправленная сеть RNN для задачи расстановки тегов POS

В случае обычной многослойной сети RNN $\mathbf{z}_t$ является скрытым вектором последнего слоя. Для двунаправленной сети RNN $\mathbf{z}_t$ обычно рассматривается как конкатенация векторов на выходе модели слева направо и модели справа налево.

Сети RNN также могут использоваться для задач классификации на уровне предложений (или на уровне документов), — в таких сетях конечный слой выдает единственное выходное значение, а не поток выходных значений, по одному на каждом временном этапе. Например, при **анализе тональности текста** цель состоит в том, чтобы классифицировать текст как выражающий *позитивное* или *негативное* мнение. Например, предложение "This movie was poorly written and poorly acted" (*Этот фильм был плохо написан и плохо поставлен*) следует классифицировать как негативное мнение. (В некоторых схемах анализа тональности текста используется более двух категорий или результатом является числовое скалярное значение.)

Использовать сети RNN для задач уровня предложения немного сложнее, так как необходимо получить совокупное представление для всего предложения $\mathbf{y}$ на основании пословных выходных значений $\mathbf{y}_t$ в сети RNN. Простейший способ сделать это — использовать скрытое состояние сети RNN, соответствующее последнему входному слову, поскольку к этому моменту сеть уже прочтет все предложение. Однако такой подход может вызвать неявное смещение модели в сторону предоставления большего внимания концу предложения. Другой распространенный метод — объединение всех скрытых векторов. Например, ►**объединение по среднему** (*average pooling*) вычисляется как поэлементное среднее по всем скрытым векторам:

$$\tilde{\mathbf{z}} = \frac{1}{s}\sum_{t=1}^{s} \mathbf{z}_t .$$

Объединенный d-мерный вектор $\tilde{\mathbf{z}}$ может быть передан в один или несколько слоев с прямой связью, прежде чем поступит в выходной слой.

24.2.3. Архитектура LSTM для задач обработки естественного языка

Уже говорилось, что сети RNN иногда позволяют решить проблему ограниченного контекста. В теории любая информация может быть передана от одного скрытого слоя к следующему для любого количества временных этапов. Но на практике информация при этом может быть потеряна или искажена, как в детской игре в испорченный телефон, когда играющие по очереди, от первого к последнему, шепчут соседу на ухо какое-либо сообщение. Как правило, то сообщение, которое получит последний играющий, будет искажено и довольно сильно отличаться от исходного. Для сетей RNN данная проблема аналогична проблеме **исчезающего градиента**, обсуждавшейся в разделе 21.1.2, — за исключением того, что здесь речь идет о состоянии слоев во времени, а не о глубоких слоях.

В разделе 21.6.2 была представлена модель **долгой краткосрочной памяти** (*long short-term memory* — LSTM). Это тип сетей RNN с вентилями, не подверженными проблеме несовершенного воспроизведения сообщения при переходе от одного временно*г*о этапа к следующему. Вместо этого модель LSTM может решить *запомнить* некоторые части ввода, копируя их на следующий этап времени, и забыть остальные части. Рассмотрим модель языка для обработки текста, подобного приведенному ниже.

> The athletes, who all won their local qualifiers and advanced to the finals in Tokyo, now...
> (*Спортсмены, которые все выиграли свои местные отборочные матчи и вышли в финал в Токио, теперь...*)

Если в данный момент спросить модель, какое следующее слово более вероятно, “compete” (*соревнуются*) или “competes” (*соревнуется*), можно ожидать, что она выберет слово “compete”, потому что оно согласуется по числу с подлежащим “The athletes”. Сеть LSTM может научиться создавать скрытый признак для лица и числа подлежащего в предложении и копировать этот признак вперед без изменений, пока не потребуется сделать выбор, подобный данному. Обычные сети RNN (или *n*-граммные модели для подобного случая) в такой ситуации часто ошибаются в длинных предложениях со многими промежуточными словами между подлежащим и глаголом.

24.3. Модели “от последовательности к последовательности”

Одним из наиболее широко изучаемых классов задач в обработке естественного языка является ►**машинный перевод** (*machine translation* — ►**MT**), целью которого является перевод предложения с ►**исходного языка** на ►**целевой язык**, например с испанского на английский. Модель машинного перевода в этом случае обучается на большом корпусе пар исходных/целевых предложений. Конечная цель обучения состоит в точном переводе новых предложений, которых не было в данных, использовавшихся при обучении.

Можно ли использовать рекуррентную нейронную сеть для создания системы MT? Безусловно, можно закодировать исходное предложение с помощью RNN. Если бы существовало однозначное соответствие между словами на исходном и целевом языках, то можно было бы рассматривать задачу машинного перевода как простую задачу расстановки тегов, — получив исходное слово “perro” на испанском языке, просто помечаем его соответствующим английским словом “dog” (*собака*). Но на самом деле слова не всегда находятся в отношении один к одному: в испанском языке три слова “caballo de mar” соответствуют одному английскому слову “seahorse” (*морской конек*), а два слова “perro grande” переводятся как “big dog” (*большая собака*) с обратным порядком слов. Изменение порядка слов может быть еще более экстремальным: в английском языке подлежащее обычно

находится в начале предложения, тогда как на языке фиджи подлежащее обычно его завершает. Так как же можно сгенерировать предложение на целевом языке?

Создается впечатление, что следует генерировать по одному слову за раз, но при этом следить за контекстом, чтобы помнить те части исходного предложения, которые еще не были переведены, и отслеживать ту его часть, которая уже была переведена, чтобы не повторяться. Также кажется, что для некоторых предложений потребуется обработать все исходное предложение, прежде чем приступать к генерированию целевого. Другими словами, генерация каждого слова целевого предложения зависит от всего исходного предложения и от всех ранее сгенерированных слов целевого предложения.

Все это придает процедуре генерации текста при машинном переводе большое сходство с работой стандартной RNN-модели языка, обсуждавшейся в разделе 24.2. Конечно, если обучать сеть RNN на текстах на английском языке, то она с большей вероятностью породит фразу “big dog” (*большая собака*), чем “dog big.” (*собака большая*). Однако в задачах MT требуется генерировать не просто любое случайное предложение на целевом языке, — на этом языке необходимо сгенерировать такое предложение, которое *соответствует* предложению исходного языка. Самый простой способ достичь этого — использовать две сети RNN: одну — для источника, а другую — для цели. В сеть RNN источника поступает исходное предложение, а затем окончательное скрытое состояние этой сети используется как начальное скрытое состояние для сети RNN цели. В результате каждое целевое слово неявно обусловливается как всем исходным предложением, так и предыдущими целевыми словами.

Такая архитектура нейронной сети называется базовой моделью ►**от последовательности к последовательности** (*sequence-to-sequence model* или сокращенно *seq2seq model*), — пример такой сети приведен на рис. 24.6. Модели от последовательности к последовательности чаще всего используются для машинного перевода, но могут использоваться и при решении ряда других задач, таких как автоматическая генерация текстовой подписи к изображению или обобщение: преобразование длинного текста в более короткий с сохранением прежнего смыслового значения.

Базовые модели от последовательности к последовательности были значительным прорывом в области обработки естественных языков в целом и машинного перевода в частности. Согласно By и соавт. ([2392], 2016) этот подход привел к уменьшению количества ошибок на 60% по сравнению с предыдущими методами машинного перевода. Однако этим моделям свойственны три серьезных недостатка.

- **Смещение к ближнему контексту.** Как бы RNN-сети не стремились сохранить знания о прошлом, эти знания должны соответствовать их скрытому состоянию. Например, пусть рекуррентная нейронная сеть обрабатывает слово (или временно*й* этап) под номером 57 в последовательности из 70 слов. Скрытое состояние, вероятно, будет содержать больше информации о слове, которое обрабатывалось на этапе 56, чем о слове, обрабатывавшемся

на этапе 5, поскольку каждый раз, когда скрытый вектор обновляется, эта операция должна сопровождаться заменой некоторого количества уже существующей информации новой информацией. Такое поведение является элементом, намеренно заложенным при разработке данной модели, и часто имеет большое значение при обработке естественного языка, так как ближайший контекст, как правило, более важен. Однако дальний контекст иногда имеет решающее значение, но может быть уже утерян в модели RNN, — даже модели LSTM испытывают затруднения в подобных ситуациях.

- **Фиксированный предел для размера контекста**. В модели перевода на базе сети RNN все исходное предложение сжимается в единый вектор скрытого состояния фиксированного размера. В сетях LSTM, используемых в современных моделях обработки естественного языка, обычно имеется около 1024 измерений, и если необходимо в этих 1024 измерениях представить предложение, скажем, из 64 слов, то на каждое слово будет отведено только 16 измерений, чего совершенно недостаточно для сложных предложений. С другой стороны, увеличение размера вектора скрытого состояния может привести к замедлению обучения и переобучению.
- **Медленная последовательная обработка**. Как указывалось в разделе 21.3, нейронные сети достигают значительного повышения эффективности за счет обработки обучающих данных, сгруппированных в пакеты, — это позволяет реализовать все преимущества эффективной аппаратной поддержки матричной арифметики. С другой стороны, сети RNN просто вынуждены работать с обучающими данными по одному слову на каждом этапе.

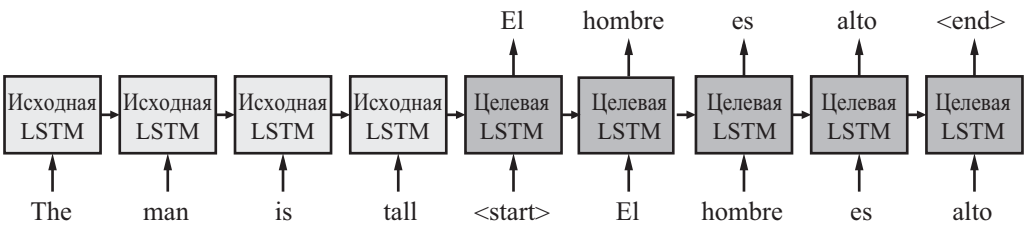


Рис. 24.6. Базовая модель от последовательности к последовательности. Каждый блок представляет один временной этап работы сети LSTM. (Для упрощения встроенный и выходной слои не показаны.) Серией последовательных этапов в сеть вводятся слова исходного предложения “The man is tall” (*Мужчина высокий*), за которыми следует тег `<start>`, указывающий, что сеть должна начать генерировать целевое предложение. Конечное скрытое состояние на этапе конца исходного предложения используется как скрытое состояние начала генерации целевого предложения. Далее каждое слово целевого предложения в момент времени t используется в качестве ввода в момент времени $t + 1$, пока сеть не выведет тег `<end>`, указывающий, что генерация предложения завершена

24.3.1. Механизм внимания

Что мы получим, если целевую сеть RNN обусловить *всеми* скрытыми векторами из исходной сети RNN, а не только последним? Это уменьшит влияние недостатков смещения к ближнему контексту и фиксированного предела для размера контекста, позволив модели с одинаковым успехом обращаться к любому предыдущему слову. Один способ достижения такого доступа заключается в объединении всех скрытых векторов RNN-источника. Однако это решение приводит к огромному увеличению количества весов с одновременным увеличением времени вычислений и, возможно, появлением переобучения. Вместо этого можно воспользоваться тем фактом, что, когда целевая сеть RNN генерирует целевые слова по одному на каждом этапе, весьма вероятно, что на самом деле лишь небольшая часть исходного вектора релевантна каждому генерируемому целевому слову.

Крайне важно понимать, что целевая сеть RNN для каждого генерируемого слова должна обращать внимание на разные части исходного вектора. Предположим, что сеть обучена переводу с английского языка на испанский. На входе получены слова “The front door is red” (*Входная дверь красная*), за которыми следует маркер конца предложения, означающий, что пора начинать вывод испанских слов. Поэтому в идеале следует сначала обратить внимание на слово “The” и сгенерировать соответствующее испанское слово “La”, а затем обратить внимание на слово “door” и сгенерировать вывод “puerta”, и т.д.

Эту концепцию можно формализовать с помощью компонента нейронной сети, называемого ►**вниманием** (*attention*), который может использоваться для создания “основанного на контексте обобщения” исходного предложения в представлении с фиксированной размерностью. Вектор контекста $\mathbf{c}_i$ содержит наиболее релевантную информацию для генерации следующего целевого слова и будет использоваться в качестве дополнительного ввода в целевой сети RNN. Модель от последовательности к последовательности, использующая компонент внимания, называется моделью ►**от последовательности к последовательности с вниманием**. Если стандартная целевая сеть RNN описывается как

$$\mathbf{h}_i = RNN(\mathbf{h}_{i-1}, \mathbf{x}_i),$$

то целевая сеть RNN для модели от последовательности к последовательности с вниманием может быть описана как

$$\mathbf{h}_i = RNN(\mathbf{h}_{i-1}, [\mathbf{x}_i; \mathbf{c}_i]),$$

где $[\mathbf{x}_i; \mathbf{c}_i]$ — конкатенация входного вектора $\mathbf{x}_i$ и контекстного вектора $\mathbf{c}_i$, определенная как

$$r_{ij} = \mathbf{h}_{i-1} \cdot \mathbf{s}_j$$
$$a_{ij} = e^{r_{ij}} / (\sum_k e^{r_{ik}})$$
$$\mathbf{c}_i = \sum_j a_{ij} \cdot \mathbf{s}_j ,$$

где $\mathbf{h}_{i-1}$ является вектором целевой сети RNN, который будет использоваться для прогнозирования слова на временно*м* этапе i, а $\mathbf{s}_j$ — выходной вектор исходной сети RNN для исходного слова (или временно*го* этапа) j. Как $\mathbf{h}_{i-1}$, так и $\mathbf{s}_j$ являются d-мерными векторами, где d — скрытый размер. Следовательно, значение r_{ij} является необработанной "оценкой внимания" между текущим целевым состоянием и исходным словом j. Затем эти оценки нормализуются в вероятность a_{ij} с использованием функции активации softmax для всех исходных слов. В завершение эти вероятности используются для генерации значения взвешенного среднего исходных векторов сети RNN $\mathbf{c}_i$ (еще один d-мерный вектор).

Пример модели от последовательности к последовательности с вниманием приведен на рис. 24.7, *а*. Здесь есть несколько важных деталей, на которых следует остановиться. Во-первых, сам компонент внимания не имеет обученных весовых коэффициентов и поддерживает последовательности переменной длины как на стороне источника, так и на стороне цели. Во-вторых, как и в большинстве других методов моделирования нейронных сетей, о которых шла речь выше, механизм внимания полностью скрыт. Программист не диктует, когда и какая информация используется; модель сама изучает, что ей следует использовать. Механизм внимания также можно комбинировать с многослойными рекуррентными нейронными сетями. В этом случае механизм внимания, как правило, применяется к каждому слою.

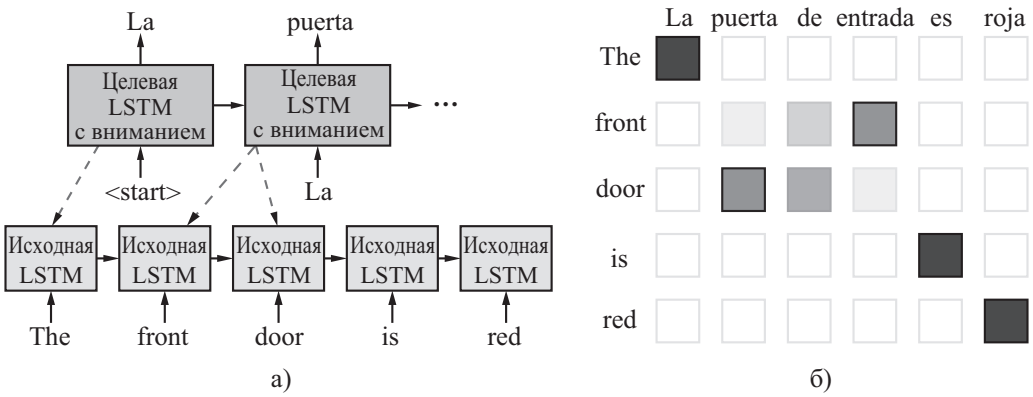


Рис. 24.7. а) Модель от последовательности к последовательности с вниманием для перевода с английского языка на испанский. Пунктирные линии представляют механизм внимания. **б)** Пример матрицы вероятностей механизма внимания для пары предложений на двух языках, — более темные прямоугольники здесь представляют более высокие значения a_{ij}. Сумма вероятностей в каждом столбце матрицы внимания равна единице

Использование в механизме внимания вероятностной формулировки с функцией активации softmax служит трем целям. Во-первых, это делает внимание функцией дифференцируемой, что необходимо для работы метода обратного распространения. Несмотря на то что сам по себе механизм внимания не имеет обучаемых весовых коэффициентов, градиенты по-прежнему возвращаются через этот механизм к исходной и целевой сетям RNN. Во-вторых, вероятностная формулировка позволяет модели захватывать определенные типы контекстуализации на больших расстояниях, которые, возможно, не были захвачены исходной сетью RNN, поскольку механизм внимания способен рассмотреть сразу всю исходную последовательность и научиться сохранять только то, что важно, игнорируя все остальное. В-третьих, вероятностный характер механизма внимания позволяет сети представлять неопределенность — если сети не известно точно, какое исходное слово переводить следующим, она может распределить вероятности внимания по нескольким параметрам, а затем в конечном итоге выбрать слово, используя целевую сеть RNN.

В отличие от большинства других компонентов нейронных сетей, вероятности механизма внимания часто могут интерпретироваться людьми и являются интуитивно значимыми. Например, в случае машинного перевода вероятности внимания часто соответствуют межсловесному выравниванию, которое мог бы выполнить человек. Это наглядно представлено на рис. 24.7, *б*.

Модели от последовательности к последовательности естественным образом подходят для машинного перевода, но почти любая задача обработки естественного языка может быть переопределена как задача, ориентирующаяся на модель этого типа. Например, система предоставления ответов на вопросы может обучаться на входных данных, состоящих из вопроса, за которым следует разделитель, за которым следует правильный ответ.

24.3.2. Декодирование

Во время обучения модели от последовательности к последовательности предпринимаются попытки максимизировать вероятность каждого слова в целевом обучающем предложении, обусловленную исходным предложением и всеми предыдущими целевыми словами. После завершения обучения модели передается исходное предложение и ее задача состоит в том, чтобы сгенерировать соответствующее целевое предложение. Как показано на рис. 24.7, можно генерировать целевое предложение по одному слову на каждом этапе, а затем возвращать его в качестве обратной связи для слова, которое генерируется на следующем этапе. Данная процедура называется ►**декодированием**.

Самая простая форма декодирования — выбирать слово с наибольшей вероятностью на каждом временно*м* этапе, а затем передавать его в качестве ввода для следующего этапа. Этот подход называется ►**жадным декодированием**, потому что после генерации каждого целевого слова система полностью соглашается

с гипотезой, которую она сгенерировала до этого момента. Проблема здесь состоит в том, что цель декодирования — максимизировать вероятность *всей* полученной целевой последовательности, чего можно и не достичь при жадном декодировании. Например, рассмотрим использование метода жадного декодирования при переводе на испанский язык предложения на английском, которое уже упоминалось выше: “The front door is red.” (*Входная дверь красная*).

Правильным переводом будет “La puerta de entrada es roja” — на английском языке буквально “The door of entry is red.” (*Дверь входа красная*). Предположим, что целевая сеть RNN правильно генерирует первое слово “La”, соответствующее артиклю “The” в исходном предложении. Далее жадный декодер может предложить слово “entrada” для исходного слова “front”, но это будет ошибкой — согласно правильному порядку слов в испанском языке существительное “puerta” должно стоять *перед* определяющим его прилагательным. Жадное декодирование работает быстро, рассматривая только один вариант выбора на каждом этапе, что требует немного времени, но эта модель не имеет механизма для исправления возможных ошибок.

Можно попробовать улучшить механизм внимания таким образом, чтобы он всегда обращал внимание на слово справа и каждый раз делал правильное предположение. Однако для многих предложений невозможно правильно угадать все слова в начале предложения, пока не станет известно, что у него в конце.

Лучшим подходом является поиск оптимального декодирования (или, по крайней мере, хорошего) с использованием одного из алгоритмов поиска, обсуждавшихся в главах 3 и 4. Чаще всего выбираемый вариант — **поиск по лучу** (см. раздел 4.1.3). В контексте процедуры декодирования при машинном переводе при поиске по лучу на каждом этапе обычно сохраняются первые k гипотез, затем каждая расширяется на одно слово, используя лучшие k вариантов слов, после чего выбираются k лучших из полученных k^2 новых гипотез. Когда все гипотезы в луче генерируют специальный токен <end>, алгоритм передает на выход гипотезу с наивысшей оценкой.

Визуализация процесса поиска по лучу при декодировании приведена на рис. 24.8. По мере того как модели глубокого обучения становятся все более и более точными, обычно появляется возможность использовать луч меньшего размера. На текущий момент в нейронных сетевых моделях машинного обучения используется луч размером от 4 до 8, тогда как в более ранних поколениях статистических моделей машинного перевода размер луча достигал 100 или более.

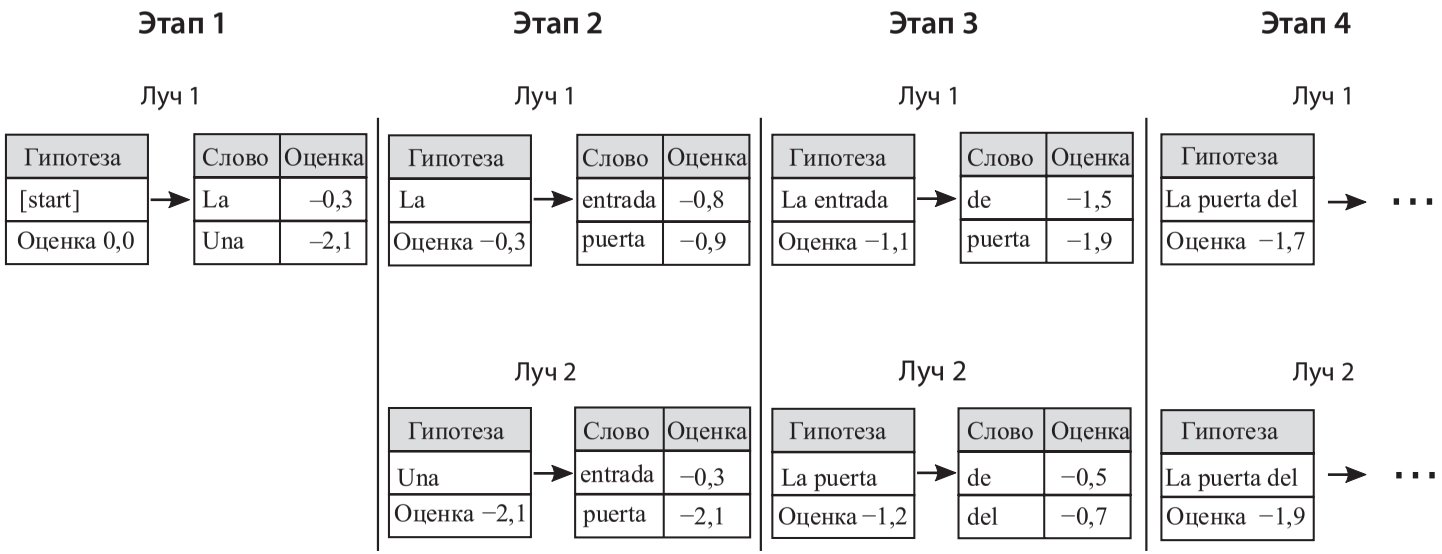


Рис. 24.8. Поиск по лучу с размером луча $b = 2$. Оценка каждого слова — это логарифм вероятности, генерируемой целевой сетью RNN с функцией активации softmax, а оценкой каждой гипотезы является сумма оценок слов. На временном этапе 3-я гипотеза с самой высокой оценкой “La entrada” позволяет генерировать продолжения только с очень небольшой вероятностью и поэтому отбрасывается — “выпадает из луча”

24.4. Архитектура "трансформер"

Во влиятельной статье "Attention is all you need" (*Внимание — это все, что вам нужно*) (Вашвани и др. [2266], 2018) впервые была представлена архитектура ►**трансформер**, в которой используется механизм ►**самовнимания**, способный моделировать дальний контекст без последовательной зависимости.

24.4.1. Механизм самовнимания

В рассмотренных выше моделях класса от последовательности к последовательности механизм внимания был направлен от целевой сети RNN к исходной сети RNN. Концепция ►**самовнимания** расширяет этот механизм таким образом, что каждая последовательность скрытых состояний применяет механизм внимания и к самой себе: исходная — к исходной, а целевая — к целевой. Это позволяет модели дополнительно захватывать удаленный (и близлежащий) контекст в каждой последовательности.

Наиболее простой способ применения механизма самовнимания — это когда матрица внимания непосредственно формируется точечным произведением входных векторов. Однако этот вариант проблематичен: точечное произведение вектора с самим собой всегда будет высоким, поэтому каждое скрытое состояние будет смещено по отношению к себе. В архитектуре "трансформер" эта проблема решается предварительным проецированием входа на три разных представления с использованием трех разных весовых матриц.

- ►**Вектор запроса** $\mathbf{q}_i = \mathbf{W}_q\mathbf{x}_i$ — это вектор, *от которого* направлено внимание, подобно целевому вектору в стандартном механизме внимания.
- ►**Вектор ключа** $\mathbf{k}_i = \mathbf{W}_k\mathbf{x}_i$ — это вектор, *на который* направлено внимание, подобно исходному вектору в базовом механизме внимания.
- ►**Вектор значения** $\mathbf{v}_i = \mathbf{W}_v\mathbf{x}_i$ — это генерируемый контекст.

В стандартном механизме внимания сети ключа и значения идентичны, но интуитивно понятно, что есть смысл сделать их отдельными представлениями. Результаты кодирования i-го слова $\mathbf{c}_i$ можно рассчитать, применив механизм внимания к проецируемым векторам:

$$r_{ij} = (\mathbf{q}_i \cdot \mathbf{k}_j)/\sqrt{d}$$

$$a_{ij} = e^{r_{ij}}/(\sum_k e^{r_{ik}})$$

$$\mathbf{c}_i = \sum_j a_{ij} \cdot \mathbf{v}_j ,$$

где d — размерность векторов $\mathbf{k}$ и $\mathbf{q}$. Обратите внимание, что i и j — это индексы в одном предложении, поскольку контекст кодируется с использованием механизма самовнимания. В каждом слое архитектуры "трансформер" механизм

самовнимания использует скрытые векторы из предыдущего слоя, который исходно является слоем векторов встраивания слов.

Имеется несколько важных деталей, о которых здесь стоит упомянуть. Во-первых, механизм самовнимания *асимметричен*, так как значение r_{ij} отличается от r_{ji}. Во-вторых, для улучшения числовой стабильности в формулу был добавлен коэффициент масштабирования $\sqrt{d}$. В-третьих, кодировку для всех слов в предложении можно рассчитать одновременно, поскольку приведенные выше уравнения могут быть записаны с использованием матричных операций, которые можно эффективно вычислять параллельно на современном специализированном оборудовании.

Выбор того, какой контекст использовать, не задается предварительно, а полностью изучается моделью на обучающих примерах. Контекстное обобщение $\mathbf{c}_i$ представляет собой сумму по всем предыдущим позициям в предложении. Теоретически в $\mathbf{c}_i$ может быть представлена и информация из предложения, но на практике важная информация иногда оказывается утерянной, поскольку она, по сути, усредняется по всему предложению. Один из способов решения этой проблемы называется ►**множественным вниманием** (*multiheaded attention*). Предложение делится на вплоть до m равных частей, и механизм внимания модели применяется к каждой из этих m частей. Каждая часть получает собственный набор весов, а затем полученные результаты конкатенируются в вектор $\mathbf{c}_i$. За счет выполнения операции конкатенации, а не суммирования выделение важных частей предложения упрощается.

24.4.2. От механизма самовнимания к архитектуре "трансформер"

Механизм самовнимания является лишь одним из компонентов модели архитектуры "трансформер". Каждый слой в модели архитектуры "трансформер" состоит из нескольких подслоев, и в каждом слое этой модели в первую очередь применяется самовнимание. Выход модуля внимания пропускается через слои с прямой связью, в которых одинаковые весовые матрицы прямой связи независимо применяются к каждой позиции. Нелинейная функция активации, обычно ReLU, применяется после первого слоя с прямой связью. Для решения проблемы исчезающего градиента в слой модели "трансформер" добавляются два остаточных соединения. На рис. 24.9 приведена структура модели "трансформер" с одним слоем. На практике модели "трансформер" обычно имеют шесть и более слоев. Как и в случае других моделей, обсуждавшихся выше, выход слоя i используется в качестве входных данных для слоя $i+1$.

В архитектуре "трансформер" порядок слов в предложении не фиксируется явно, поскольку контекст моделируется только через механизм самовнимания, который безразличен к порядку слов. Чтобы захватить порядок слов, в модели "трансформер" используется метод, получивший название ►**позиционное**

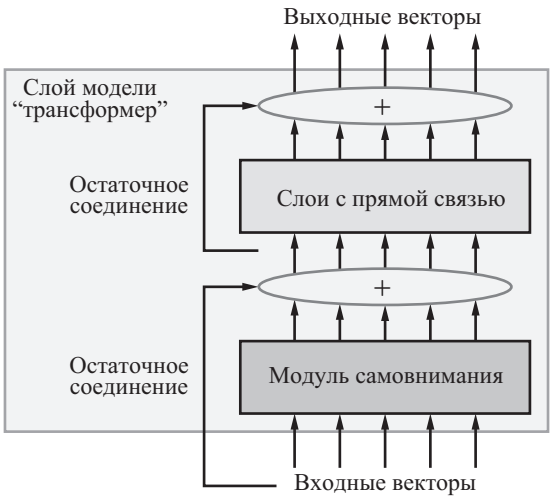


Рис. 24.9. Модель "трансформер" с одним слоем, состоящим из сети с прямой связью и остаточных связей

встраивание. Если входная последовательность имеет максимальную длину n, то изучается n новых векторов встраивания — по одному на каждую позицию слова и вход в первый слой модели "трансформер" представляет собой сумму встраивания слова в позиции t плюс позиционное встраивание, соответствующее позиции t.

На рис. 24.10 представлена схема модели архитектуры "трансформер" для задачи проставления тегов частей речи (POS), на вход которой подано то же самое предложение, что и на рис. 24.3. В нижней части схемы значения встраивания слов и позиционное встраивание суммируются с целью формирования входного вектора для модели "трансформер" с тремя слоями. Модель "трансформер" генерирует по отдельному вектору для каждого слова, как и в случае модели для расстановки тегов POS на основе сети RNN. Каждый вектор слова подается в последний выходной слой с функцией активации softmax для получения распределения вероятности по тегам.

В этом разделе фактически была рассказана только половина истории под названием "архитектура трансформер": та модель, которая была описана выше, в действительности называется ►**энкодером трансформера**. Она может быть полезна для решения задач классификации текста. Полная архитектура "трансформер" изначально была разработана как улучшенный вариант модели от последовательности к последовательности для задач машинного перевода. Поэтому в дополнение к энкодеру эта архитектура включает в себя ►**декодер трансформера**. Кодер и декодер почти идентичны, — за исключением того, что в декодере

используется такая версия модуля самовнимания, в которой каждое слово может соотноситься только с предыдущими словами, так как текст генерируется слева направо. Декодер также имеет второй модуль внимания в каждом слое архитектуры “трансформер”, который связан с выходным вектором энкодера.

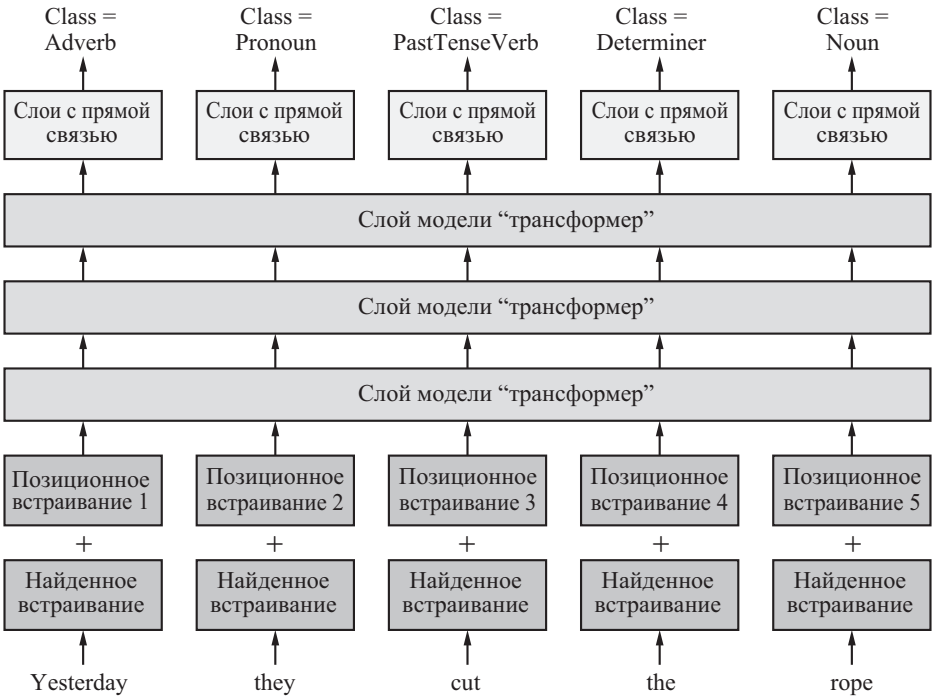


Рис. 24.10. Использование архитектуры “трансформер” применительно к задаче расстановки тегов частей речи

24.5. Предобучение и трансферное обучение

Получение достаточного количества данных для построения надежной модели может оказаться серьезной проблемой. В области компьютерного зрения (см. главу 25) эта проблема была решена посредством создания больших коллекций изображений (таких, как ImageNet) и ручной их маркировки.

В области обработки естественного языка более распространенным вариантом является работа с текстом без маркировки. Эта особенность отчасти связана с трудностью подобной маркировки: неквалифицированному работнику несложно присвоить изображению метку “кошка” или “закат”, но, чтобы аннотировать предложение тегами частей речи или построить дерево его синтаксического анализа,

необходим работник, прошедший серьезную подготовку. Другое отличие связано с обилием текста: в Интернете каждый день добавляется более 100 млрд слов текста, включая оцифрованные книги, редактируемые ресурсы, подобные Википедии, и не редактируемые посты в социальных сетях.

Такие проекты, как Common Crawl, предоставляют легкий доступ к подобным данным. Любой последовательный текст может использоваться для построения *n*-грамм или моделей встраивания слов, а некоторые тексты предоставляются со структурой, которая может быть полезна для решения различных задач, например есть много сайтов, содержащих ответы на часто задаваемые вопросы, т.е. пары "вопрос–ответ", которые можно использовать для обучения системы предоставления ответов на вопросы. Аналогичным образом многие веб-сайты публикуют тексты с параллельным переводом, которые можно использовать для обучения системы машинного перевода. Некоторые тексты предоставляются даже с определенными метками, — например, на сайтах с различными обзорами пользователи могут комментировать текст обзора с использованием 5-звездочной рейтинговой системы.

Было бы очень желательно избавиться от необходимости создавать новый набор данных всякий раз, когда потребуется построить новую модель обработки естественного языка. В этом разделе вводится идея ►**предобучения** — формы **трансферного обучения** (см. раздел 21.7.2), в которой используется большое количество общедоступных языковых данных общего характерах для обучения начальной версии модели обработки естественного языка. Достигнув этой отправной точки, для уточнения модели можно будет использовать уже меньший объем специализированных данных (возможно, включая некоторые помеченные данные), благодаря чему уточненная модель сможет быстро выучить дополнительный словарный запас, идиомы, синтаксические структуры и другие лингвистические явления, специфичные для данной конкретной области.

24.5.1. Предобученные встраивания слов

Выше, в разделе 24.1, кратко обсуждалась концепция встраивания слов. Было показано, что в этом случае схожие слова — скажем, *банан* и *яблоко* — в конечном итоге представляются сходными векторами, а задачу поиска аналогии оказывается возможным решить посредством простого векторного вычитания. Все это указывает на то, что метод встраивания слов позволяет выявить и зафиксировать важную информацию о словах.

В этом разделе будет детально рассмотрен вопрос о том, как встраивания слов создаются с использованием полностью неконтролируемого процесса обработки большого корпуса текста. Этот подход совершенно отличается от процедуры, описанной в разделе 24.1, где встраивания создавались в процессе обучения с учителем на основе данных с метками частей речи, расстановка которых требовала дорогостоящего ручного аннотирования текстов.

Здесь будет рассмотрена лишь одна конкретная модель построения встраивания слов — модель GloVe (Global Vectors). Она начинает работу со сбора счетчиков количества появления каждого слова в окне другого слова, что похоже на модель скип-граммы. Сначала выбирается размер окна (допустим, 5 слов), и пусть X_{ij} будет количеством раз, когда слова i и j одновременно присутствуют внутри окна, а X_i пусть будет количеством раз, когда слово i встречается в окне с любым другим словом. Обозначим через $P_{ij} = X_{ij} / X_i$ вероятность появления слова j в контексте слова i. Наконец, как и прежде, пусть $\mathbf{E}_i$ — это вектор встраивания для слова i.

Одно из принятых в модели GloVe допущений состоит в том, что лучше всего связь между двумя словами можно выявить, сравнив оба эти слова с другими словами. В качестве примера возьмем слова “ice” (*лед*) и “steam” (*пар*). Рассмотрим отношение вероятностей их совместного появления в окне с другим словом w, т.е.

$$P_{w,ice} / P_{w,steam} .$$

Если w — слово “solid” (*твердый*), это отношение будет большим (это означает, что понятие *твердый* в большей степени относится к слову *лед*), а когда w — слово “gas” (*газ*), оно будет невелико (и это означает, что слово *газ* ближе к слову *пар*). Если же w является словом, не имеющим собственного значения, подобно артиклю “the”, или таким словом, как “water” (*вода*), которое одинаково релевантно к обоим, или одинаково нерелевантно к ним как слово “fashion” (*мода*), то это отношение будет близко к единице.

Отталкиваясь от этого интуитивного допущения, модели GloVe было предоставлено некоторое математическое обоснование (Пеннингтон и др. [1772], 2014), в котором отношения вероятностей преобразуются в векторные разности и точечные произведения и в конечном счете сводятся к ограничению

$$\mathbf{E}_i \cdot \mathbf{E}'_k = \log(P_{ij}).$$

Другими словами, скалярное произведение двух векторов встраиваний слов равно логарифму вероятности их совместного появления в окне. Это утверждение интуитивно понятно: два почти перпендикулярных друг другу вектора будут иметь точечное произведение, близкое к нулю, а два почти идентичных нормализованных вектора будут иметь точечное произведение, близкое к единице. Здесь есть одно техническое усложнение: модель GloVe создает два вектора встраивания слов для каждого слова: $\mathbf{E}_i$ и E'_i — вычисление двух векторов с последующим их сложением в конечном счете способствует ограничению переобучения.

Обучение такой модели, как GloVe, как правило, обходится намного дешевле, чем обучение стандартной нейронной сети: новая модель может быть обучена на миллиардах слов текста за несколько часов при использовании стандартного настольного компьютера.

Можно обучить предобученные векторы встраивания слов для конкретной проблемной области, а затем извлекать из модели знания по ней. Например, Цитоян

и соавт. ([2228], 2019) использовали 3,3 млн научных рефератов по теме материаловедения для обучения модели встраивания слов. Они установили, что, подобно тому, как общая модель встраивания слов может ответить на вопрос “Афины для Греции, как Осло для [что?]” словом “Норвегии”, их обученная материаловедению модель на вопрос “NiFe — ферромагнетик, а IrMn — это [что?]” может дать ответ “антиферромагнетик”.

Их модель не полагается исключительно на совместное появление слов в окне, — создается впечатление, что она способна выделять и фиксировать более сложные научные знания. На вопрос, какие химические соединения можно классифицировать как “термоэлектрик” или “топологический изолятор”, эта модель оказалась способной ответить правильно. Однако соединение $CsAgGa_2Se_4$ в обучающем корпусе никогда не появляется в окне вместе со словом “термоэлектрик”, но появляется вместе со словами “халькогенид”, “запрещенная зона” и “оптоэлектрик”, которые являются ключами, позволяющими классифицировать это соединение как “термоэлектрик”. Более того, при обучении только на рефератах, относящихся к 2008 году, и при выдаче запроса отобрать соединения, которые являются “термоэлектриками”, но еще не упоминаются как таковые в рефератах, для трех из пяти лучших пиков, предложенных моделью, было обнаружено, что они действительно описываются как термоэлектрики в работах, опубликованных между 2009 и 2019 годами.

24.5.2. Предобученные контекстные представления

Встраивания слов являются лучшими представлениями слов, чем атомарные токены, но здесь имеет место важная проблема: многозначность слов. Например, в английском языке слово “rose” (*роза*) может являться названием цветка или быть формой прошедшего времени глагола “rise” (*подниматься*). Следовательно, для “rose” можно ожидать обнаружить по крайней мере два совершенно разных кластера близких по значению слов: один будет соответствовать названиям цветов, таким как “dahlia” (*георгина*), а другой — глаголам, определяющим состояние подъема, таким как “upsurge” (*повышаться*). Ни один вектор встраивания не сможет захватить оба эти контекста одновременно. Слово “rose” является ярким примером слов с (по крайней мере) двумя совершенно различными значениями, но другие слова могут обладать более тонкими оттенками значения, зависящими от контекста, например слово “need” (*нужно*, *требуется*) в фразах “you need to see this movie” (*тебе нужно посмотреть этот фил*ьм) и “humans need oxygen to survive” (*людям требуется кислород, чтобы жить*). А некоторые идиоматические фразы, такие как “break the bank” (*сорвать банк*), вообще лучше анализировать как единое целое, а не по входящим в него словам.

Поэтому, вместо того чтобы просто обучать таблицу “слово–встраивание”, желательно обучить модель создавать ►**контекстные представления** для каждого слова в предложении. Контекстное представление отображает в вектор

встраивания как само слово, так и слова из окружающего его контекста. Иначе говоря, если передать в эту модель слово “rose” в контексте “the gardener planted a rose bush” (*садовник посадил розовый куст*), она должна создать контекстное встраивание, которое будет похоже (но не обязательно идентично) представлению, которое будет получено для него в контексте “the cabbage rose had an unusual fragrance” (*капустная роза имеет необычный аромат*) и очень отличается от представления для слова “rose” в контексте “the river rose five feet” (*река поднялась на пять футов*).

На рис. 24.11 показана рекуррентная сеть, создающая контекстные встраивания слов, — на рисунке они представлены прямоугольниками без обозначений. Предполагается, что коллекция неконтекстных встраиваний слов уже собрана. На вход сети подается предложение “The red car is big.” (*Этот красный автомобиль большой*) по одному слову за раз, и модель должна дать прогноз для следующего слова. Поэтому, например, на рисунке в точке, где было достигнуто слово “car”, в узел RNN на этом временно*м* этапе поступают два входа: неконтекстное встраивание для слова “car” (*автомобиль*) и контекст, кодирующий информацию из предыдущих слов “The red” (*этот красный*). На этом основании узел RNN выведет контекстное представление для слова “car”. Затем сеть как целое выведет свой прогноз для следующего слова: “is” (*есть*, *является*) и весовые коэффициенты в сети будут обновлены так, чтобы минимизировать ошибку между предсказанием и фактическим следующим словом.

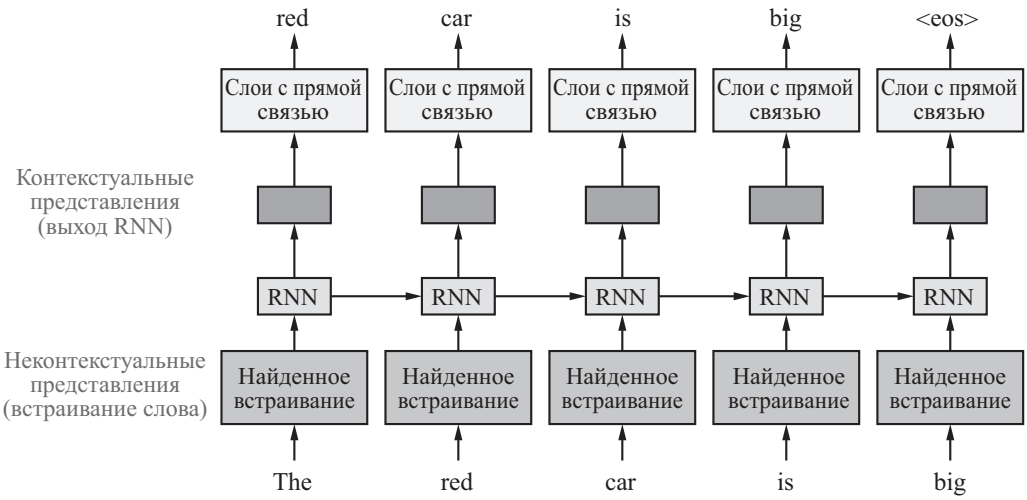


Рис. 24.11. Обучение контекстных представлений с использованием модели языка слева направо

Эта модель аналогична модели для расстановки тегов частей речи, приведенной на рис. 24.5, но с двумя важными отличиями. Во-первых, эта модель является однонаправленной (слева-направо), тогда как модель на рис. 24.5 — двунаправленной. Во-вторых, вместо предсказания тегов POS для *текущего* слова эта модель прогнозирует *следующее* слово предложения, используя предыдущий контекст. Как только модель будет построена, ее можно будет использовать для извлечения представления слов и передачи их какому-то другому заданию, — нет необходимости продолжать прогнозировать следующее слово. Обратите внимание, что для вычисления контекстных представлений всегда требуется два входа, текущее слово и контекст.

24.5.3. Маскированные модели языка

Слабость стандартных языковых моделей, таких как *n*-граммные модели, заключается в том, что определение контекста каждого слова построено только на предыдущих словах предложения. Предсказания делаются в направлении слева направо. Но иногда важный контекст присутствует в предложении позже, например слово “feet” в фразе “…rose five feet” (…поднялась на пять футов) помогает прояснить значение слов, стоящих до него.

Одним из простых обходных путей является обучение отдельной модели языка справа-налево, определяющей контекст для каждого слова на основе последующих слов в предложении, с последующей конкатенацией представлений слева направо и справа налево. Однако такая модель не способна комбинировать свидетельства из обоих направлений.

Вместо этого можно использовать ►**маскированную модель языка** (*masked language model* — ►**MLM**). Модели MLM обучают маскированием (сокрытием) отдельных слов на входе и предложением модели дать прогноз для замаскированных слов на выходе. В этой задаче можно использовать глубокие двунаправленные сети RNN или архитектуру “трансформер” поверх маскированного предложения. Например, при заданном входном предложении “The river rose five feet” (*река поднялась на пять футов*) можно замаскировать его среднее слово и получить предложение “The river __ five feet”, а затем предложить модели заполнить пропуск (рис. 24.12).

Финальные скрытые векторы, соответствующие маскированным токенам, затем используются для прогнозирования слов, которые были замаскированы, в этом примере — слова “rose” (*поднялась*). В процессе обучения одно предложение можно использовать несколько раз с разными замаскированными словами. Прелесть этого подхода прежде всего в том, что он не требует помеченных данных, — предложение предоставляет собственную метку для замаскированного слова. Если эту модель обучить на большом корпусе текста, она будет генерировать предобученные представления, позволяющие хорошо справляться с широким спектром задач обработки естественного языка (машинный перевод, предоставление ответов на вопросы, обобщение, определение авторства и т.д.).

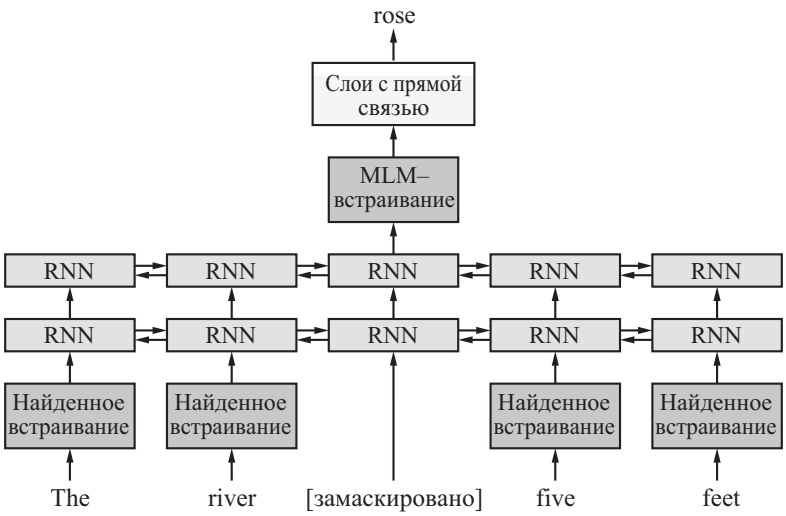


Рис. 24.12. Маскированная модель языка: предобучение двунаправленной модели — например, многослойной сети RNN — посредством маскирования входных слов и предоставления прогнозов только для этих замаскированных слов

24.6. Современное состояние разработок

Широкое применение методов глубокого и трансферного обучения способствовало заметному продвижению в современном состоянии разработок в области обработки естественного языка, причем настолько, что один комментатор в 2018 году заявил, что "пришло время ImageNet для области обработки естественных языков" (Рудер [1930], 2018). Имеется в виду, что так же, как в области компьютерного зрения, где переломный момент наступил в 2012 году, когда системы глубокого обучения дали удивительно хорошие результаты в конкурсе ImageNet, в 2018 году такой же переломный момент наступил и в области обработки естественных языков. Основной импульс для наступления этого переломного момента дало обнаружение того факта, что трансферное обучение хорошо работает в задачах обработки естественного языка: можно загрузить общую модель языка, а затем настроить ее для решения любой конкретной задачи.

Все это началось с простых встраиваний слов из таких систем, как WORD2VEC в 2013 году и GLOVE в 2014 году. Исследователи могли скачать такую модель или обучить собственную относительно быстро и без доступа к суперкомпьютерам, тогда как создание собственных предобученых контекстных представлений обходилось на порядки дороже.

Реализация этих моделей стала возможной только после того, как новые аппаратные достижения (графические процессоры и TPU) получили достаточно широкое распространение. Именно благодаря этому у исследователей появилась возможность загружать модели вместо того, чтобы тратить ресурсы на обучение собственных. Разработка архитектуры “трансформер” обеспечила возможность эффективного обучения гораздо более крупных и глубоких нейронных сетей, чем это было возможно ранее (на этот раз благодаря достижениям в области программного обеспечения, а не аппаратных средств). С 2018 года новые проекты обработки естественного языка, как правило, начинались с уже имеющейся, предобученной модели “трансформер”.

Хотя эти модели архитектуры “трансформер” обучались предсказывать следующее слово в тексте, они показали удивительно хорошие результаты и при решении других языковых задач. Модель RoBERTa при некоторой тонкой настройке демонстрирует современный уровень результатов в ответах на вопросы и тестах на понимание прочитанного (Лю и др. [1431], 2019). Модель GPT-2, с архитектурой, подобной архитектуре “трансформер” — модель языка с 1,5 млрд параметров, обученных на 40 Гбайт текста из Интернета, — показала хорошие результаты в таких разнообразных задачах, как перевод с французского языка на английский и обратно, поиск соответствий при дальних зависимостях и предоставление ответов на вопросы из области общих знаний, и все это без тонкой подстройки под конкретную задачу. Как показано на рис. 24.13, система GPT-2 способна генерировать достаточно убедительный текст, исходя всего из нескольких слов, заданных в качестве подсказки.

Дословный перевод текстов, приведенных на рис. 24.13, следующий.

Это общепризнанная истина, что земля не является центром вселенной**.** Есть те, кто утверждает это так. Я не принимаю их, но другие, я считаю, придерживаются того же мнения. Правда, однако, что если есть другое, чем центр, и если есть какие-либо другие живые существа во вселенной, и если они не люди, то у нас еще нет наших ответов. Мы должны идти дальше. Эта страница дает упрощенный, упрощенный ответ на проблему. У нас нет всех ответов. Истина в том, однако, что истина не здесь.

Когда Грегор Самса проснулся однажды утром, он не заметил ничего странного. “Когда моя жена смотрит на меня, я чувствую, что она смотрит на произведение искусства”, — сказал он. “Я думаю, она восхищается чем то, что я создал”. Идея состоит в том, что, глядя на свою жизнь, ты учишь что-то важное и становишься лучше. Это теория, которая возникла из работы психолога Даниэля Големана, в которой он спросил “Откуда ты знаешь, что ты не неудачник?”

Алиса начинала очень уставать от сидения со своей сестрой на берегу. Она села, зевнула и сказала с громким коротким криком: “Я надеюсь, ты не возражаешь, если я продолжу делать то, что я бы хотела сделать, и если кто-то спросит

меня, кто из нас будет делать больше, не говори им, что я не буду делать много, моя дорогая сестра".

Все счастливые семьи одинаковы; каждая счастливая семья похожа на райский сад. Единственная разница между счастливыми и несчастными семьями в том, что несчастные семьи не имеют никаких цветов или деревьев.

Расскажи мне историю. Расскажи мне историю. Расскажи мне историю. Расскажи мне историю. Расскажи мне историю. Расскажи мне историю. Расскажи мне историю. Расскажи мне историю. Расскажи мне историю. Расскажи мне историю. Расскажи мне историю. Пожалуйста, заполните следующие подробности. Спасибо… Спасибо за ваш интерес к этому интервью. Пожалуйста, подождите…

It is a truth universally acknowledged that the earth is not the center of the universe. There are those who assert there is. I do not accept them, but others I consider to be of the same opinion. The truth is, however, that if there are other than the center, and if there are any other living things in the universe and if they are not human, then we do not yet have our answers. We have to go on. This page gives a simplified, simplified answer to the problem. We don't have all the answers. The truth is, however, that the truth is out there.

When Gregor Samsa woke up one morning, he did not notice anything strange. "When my wife is looking at me, I feel like she is looking at a piece of art," he said. "I think she is admiring something I have created." The idea is that by looking at your own life, you learn something important and become a better person. It is a theory that emerged from psychologist Daniel Goleman's work, in which he asked "How do you know you're not a loser?"

Alice was beginning to get very tired of sitting with her sister on the bank. She sat up, yawned, and said, with a loud little scream, "I hope you don't mind if I keep on doing what I should like to do, and if someone asks me which of us will do more, don't tell them that I won't do much, my dear sister."

All happy families are alike; each happy family is like a garden of paradise. The only difference between happy families and unhappy families, is that the unhappy family doesn't have any flowers or trees.

Tell me a story. Tell me a story. Tell me a story. Tell me a story. Tell me a story. Tell me a story. Tell me a story. Tell me a story. Tell me a story. Tell me a story. Tell me a story. Please fill out the following details. Thank you... Thank you for your interest in this interview. Please wait...

Рис. 24.13. Примеры завершенных текстов, сгенерированных моделью языка GPT-2, на основании подсказок, выделенных жирным шрифтом. Бо*л*ьшая часть этих текстов — достаточно свободный английский, по крайней мере локально. Последний пример демонстрирует, что иногда работа модели все же нарушается

Как пример современной системы обработки естественного языка можно привести систему ARISTO (Кларк и др. [447], 2019), достигшую уровня 91,6% правильных ответов на экзамене за 8-й класс по естествознанию с несколькими вариантами ответов (рис. 24.14). Система ARISTO состоит из ансамбля решателей: в некоторых используется поиск информации (подобно поисковой системе в Интернете), некоторые выполняют анализ семантического следования и проводят качественные рассуждения, а некоторые используют модель языка ROBERTA с архитектурой "трансформер". Как оказалось, сама по себе система ROBERTA на этом тестовом наборе показала результат 88,2%. Система ARISTO также достигла уровня 83% правильных ответов на более сложном экзамене за 12-й класс. (Оценка 65% считается показателем "соответствует стандартам", а оценка 85% — показателем "соответствует стандартам с отличием".)

1. **What will best separate a mixture of iron filings and black pepper?**
(*С помощью чего проще всего разделить смесь из железных опилок и черного перца?*)
a) magnet b) filter paper c) triple beam balance d) voltmeter
(1. *Магнит* 2. *Фильтровальная бумага* 3. *Точные весы* 4. *Вольтметр*)

2. **Which form of energy is produced when a rubber band vibrates?**
(*Какая форма энергии производится, когда резиновая лента вибрирует?*)
(a) chemical (b) light (c) electrical (d) sound
(1. *Химическая* 2. *Световая* 3. *Электрическая* 4. *Звуковая*)

3. **Because copper is a metal, it is**
(*Поскольку медь — это металл, она…*)
(a) liquid at room temperature (b) nonreactive with other substances
(1. *Жидкая при комнатной температуре* 2. *Не реагирует с другими веществами*)
(c) a poor conductor of electricity (d) a good conductor of heat
(3. *Плохо проводит электричество* 4. *Хорошо проводит тепло*)

4. **Which process in an apple tree primarily results from cell division?**
(*Какой процесс в яблоне является прямым результатом деления клеток?*)
(a) growth (b) photosynthesis (c) gas exchange (d) waste removal
(1. *Рост* 2. *Фотосинтез* 3. *Газообмен* 4. *Выделение отходов*)

Рис. 24.14. Экзаменационные вопросы за 8-й класс по естествознанию, на которые система ARISTO способна дать правильный ответ, используя ансамбль методов, наиболее влиятельным из которых является модель языка ROBERTA. Для ответа на эти вопросы требуются знание естественного языка, понимание структуры тестов с множественным выбором, наличие здравого смысла и некоторые научные сведения

Система ARISTO имеет некоторые ограничения. Она работает только с вопросами с несколькими вариантами ответов, а не с произвольными вопросами, и не может ни читать, ни генерировать диаграммы.[1]

Система T5 (*Text-To-Text Transfer Transformer*) предназначена для выдачи текстовых ответов на различные виды текстового ввода. Она включает в себя стандартную модель архитектуры "трансформер" "кодер-декодер", предобученную на 35 млрд слов из корпуса Colossal Clean Crawled Corpus (C4) объемом 750 Гбайт. Этот обучающий набор без маркировки был разработан для предоставления модели таких обобщаемых лингвистических знаний, которые будут полезны для решения многих конкретных задач. Затем для каждой отдельной задачи система T5 обучалась на входах, состоящих из имени задачи, отделенного двоеточием от некоторого контента. Например, при получении входа "translate English to German: That is good" (*перевести с английского на немецкий: Это хорошо*) система должна была предоставить выход "Das ist gut". Для некоторых задач вход мог быть размечен, — например, в задании по применению схемы Винограда во входных данных выделяется местоимение с неоднозначной связью. Так, для входа "referent: The city councilmen refused the demonstrators a permit because *they* feared violence" (*ссылка: "Члены городского совета отказали демонстрантам в разрешении, потому что* они *боялись насилия"*) правильным ответом будет "The city councilmen" (*члены городского совета*), а не "the demonstrators" (*демонстранты*).

Многое еще предстоит сделать для улучшения систем обработки естественного языка. Одна проблема заключается в том, что модели архитектуры "трансформер" опираются только на узкий контекст, ограниченный несколькими сотнями слов. В некоторых экспериментальных подходах предпринимаются попытки расширить этот контекст: система Reformer (Китаев и др. [1232], 2020) способна обрабатывать контекст до миллиона слов.

Недавно достигнутые результаты показывают, что использование большего количества обучающих данных приводит к получению лучших моделей, — например, система RoBERTa достигла современного уровня результатов после обучения на 2,2 трлн слов. Если использование большего количества текстовых данных позволяет достичь лучших результатов, то что произойдет, если подключить к обучению и другие типы данных: структурированные базы данных, числовые данные, изображения и видео? Для обучения на большом корпусе видеоданных необходим прорыв в скорости обработки таких данных аппаратным обеспечением, а также может потребоваться несколько новых прорывов в области теории ИИ.

Внимательный читатель может задаться вопросом, зачем в предыдущей главе рассматривались такие аспекты языка, как грамматика, синтаксический анализ и семантическая интерпретация, если в этой главе предлагается отбросить их в

[1] Было отмечено, что на некоторых экзаменах с предоставленными вариантами ответов можно получить хороший балл, даже не глядя на вопросы, поскольку у неправильных ответов есть явные признаки (Гуруранган и др. [938], 2018). Похоже, что это замечание верно и для вопросов с визуальными ответами (Чао и др. [391], 2018).

пользу моделей, управляемых исключительно данными? В настоящее время ответ на этот вопрос прост: модели, управляемые данными, легче разрабатывать и поддерживать, и на стандартных тестах их показатели лучше, чем показатели систем ручной сборки, которые можно построить с использованием разумного количества человеческих усилий на основании подходов, обсуждавшихся в главе 23. Может оказаться так, что модели архитектуры "трансформер" и родственные им изучают скрытые представления, отражающие те же самые основные идеи, что и грамматики и семантическая информация, или же внутри этих огромных моделей происходит нечто совершенно иное, — мы просто этого не знаем. Нам точно известно, что систему, обученную на текстовых данных, легче поддерживать и адаптировать к новым предметным областям и новым естественным языкам, чем такую систему, которая опирается на созданные вручную признаки.

Может случиться и так, что будущие прорывы в явном грамматическом и семантическом моделировании заставят маятник качнуться назад. Возможно, более вероятно появление гибридных подходов, объединяющих лучшие концепции из обоих направлений. Например, Китаев и Кляйн ([1233], 2018) использовали механизм внимания для улучшения традиционного синтаксического анализатора, получив наилучшие результаты среди когда-либо зарегистрированных для тестового набора Penn Treebank. Аналогичным образом Ринггаард и соавт. ([1884], 2017) показали, как можно улучшить анализатор зависимостей за счет применения встраиваний слов и рекуррентной нейронной сети. Их система SLING помещает результаты анализа непосредственно в представление семантической структуры, смягчая этим проблему накопления ошибок, неизбежную в случае традиционных конвейерных систем.

Безусловно, есть достаточно места для улучшений: системы обработки естественных языков не только по-прежнему отстают от человека при решении многих задач. Гораздо важнее то, что они, к сожалению, показывают такие результаты после обработки в тысячи раз большего объема текста, чем любой человек сможет прочитать за всю свою жизнь. Это указывает на то, что имеется еще много возможностей для новых идей от лингвистов, психологов и исследователей в области обработки естественных языков.

Резюме

Вот перечень основных тем, рассмотренных в этой главе.

- Непрерывные представления слов с использованием метода встраивания более устойчивы, чем дискретные атомарные представления, и могут быть предобучены с использованием непомеченых текстовых данных.
- Рекуррентные нейронные сети позволяют эффективно моделировать локальный и удаленный контексты, сохраняя соответствующую информацию в своих векторах скрытого состояния.

- Модели класса от последовательности к последовательности могут использоваться для задач машинного перевода и генерации текста.
- Модели архитектуры “трансформер” используют механизм самовнимания, благодаря чему они способны моделировать дальний контекст так же хорошо, как и локальный. Эти модели позволяют эффективно использовать аппаратные средства матричного умножения.
- Трансферное обучение, включающее в себя предобученные контекстные встраивания слов, позволяет разрабатывать модели на основе очень больших непомеченых корпусов текста, которые затем могут успешно применяться для решения ряда задач. Модели, предобученные для прогнозирования пропущенных слов, также могут применяться при решении других задач — включая предоставление ответов на вопросы и распознавание семантического следования — после соответствующей точной настройки для целевой предметной области.

Библиографические и исторические заметки

Распределение слов и фраз на естественном языке следует **закону Ципфа** (Ципф [2449], 1935; [2450], 1949): частота *n*-го наиболее употребляемого слова приблизительно обратно пропорциональна *n*. Это означает, что во всех системах имеет место проблема разреженности данных: даже при миллиардах слов обучающих данных неизбежно будут постоянно появляться новые слова и фразы, которые ранее системе не встречались.

Обобщению вновь появившихся слов и фраз способствуют представления, отражающие следующую основную идею: слова со схожим значением появляются в сходных контекстах. Дирвестер и соавт. ([591], 1990) предложили проецировать слова на низкоразмерные векторы посредством декомпозиции матрицы совместного появления, образованной словами и документами, в которых эти слова встречаются. Еще одна возможность заключается в обработке окружающих слов как контекста — скажем, при выборе окна размером в 5 слов. Браун и соавт. ([323], 1992) группировали слова в иерархические кластеры в соответствии с их биграмным контекстом. Подобный подход доказал свою эффективность для таких задач, как распознавание именованных объектов (Туриан и др. [2232], 2010). Систему WORD2VEC (Миколов и др. [1568], 2013) можно считать первой заметной демонстрацией преимуществ метода встраивания слов, полученных за счет обучения нейронных сетей. Векторы встраивания слов GloVe (Пеннингтон и др. [1772], 2014) были получены при непосредственной обработке матрицы совместного появления слов, построенной на основе миллиардов слов текста. Леви и Гольдберг в работе [1398] (2014) подробно объясняют, почему и как эти встраивания слов способны улавливать языковые закономерности.

Бенжио и др. ([173], 2003) впервые применили нейронные сети для построения моделей языка, предложив объединить "*а*) распределенное представление для каждого слова вместе с *б*) функцией вероятности для последовательностей слов, выраженных в терминах этих представлений". Миколов и соавт. в работе [1569] (2010) продемонстрировали использование рекуррентных нейронных сетей для моделирования локального контекста в моделях языка. Юзефович и соавт. ([1155], 2016) показали, что рекуррентная нейронная сеть (RNN), обученная на миллиарде слов, может превзойти по производительности тщательно разработанные вручную *n*-граммные модели. Важность контекстных представлений для слов была подчеркнута Петерсом и соавт. ([1780], 2018), предложившими для этих представлений особое название: ELMo (Embeddings from Language Models — *встраивания из моделей языка*).

Обратите внимание, что некоторые авторы сравнивают качество моделей языка, измеряя их ►**перплексию** (*perplexity*). Перплексия распределения вероятностей равна 2^H, где H — энтропия этого распределения (см. раздел 19.3.3). Языковая модель с меньшей перплексией будет лучшей моделью при равных прочих характеристиках. Но на практике все остальные характеристики редко бывают равными. Поэтому более информативно измерять производительность на реальных задачах, а не полагаться на перплексию.

Говард и Рудер ([1072], 2018) описывают среду разработки ULMFiT (*Universal Language Model Fine-tuning*), упрощающую тонкую настройку предобученной модели языка без необходимости использования обширного корпуса документов из целевой проблемной области. Позднее Рудер и соавт. выпустили учебное пособие ([1931], 2019) по трансферному обучению для задач обработки естественного языка.

Миколов и соавт. ([1569], 2010) выдвинули идею использования рекуррентных нейронных сетей для задач обработки естественного языка, а Суцкевер и соавт. ([2154], 2015) предложили идею глубокого обучения сетевой модели класса от последовательности к последовательности. Чжу и соавт. ([2442], 2017) и Лю и соавт. ([1429], 2018) показали, что подход с использованием метода обучения без учителя действительно работает, что существенно упрощает сбор необходимых данных. Вскоре было установлено, что такие модели могут на удивление хорошо справляться с различными задачами, например с генерацией подписей к изображениям (Карпати и Фей-Фей [1191], 2015; Виньялс и др. [2275], 2017).

Девлин и соавт. ([614], 2018) показали, что модели архитектуры "трансформер", предобученные на задачах с применением маскированной модели языка, можно непосредственно использовать для решения многих задач. Эти модели получили название "BERT" (*Bidirectional Encoder Representations from Transformers*). Предобученные модели BERT можно подвергнуть точной настройке для использования в определенных предметных областях и решения конкретных задач, включая предоставление ответов на вопросы, именование распознаваемых объектов, классификацию текстов, анализ тональности текста и логический вывод на естественном языке.

Система XLNet (Янг и др. [2399], 2019) является улучшенной моделью BERT, в которой устранены расхождения между предобучением и тонкой настройкой. Среда разработки Ernie 2.0 (Сан и др. [2148], 2019) позволяет извлечь из обучающих данных больше за счет учета порядка предложений и наличия именованных объектов, а не просто анализа совместного появления слов, — она продемонстрировала лучшую производительность в сравнении с моделями BERT и XLNet. В ответ исследователи пересмотрели и усовершенствовали модели BERT: система RoBERTa (Лю и др. [1431], 2019) использует больше данных, различных гиперпараметров и процедур обучения, что позволило ей достичь соответствия показателям XLNet. В системе Reformer (Китаев и др. [1232], 2020) диапазон контекста существенно расширен и может охватывать вплоть до миллиона слов. В то же время в системе ALBERT (*A Lite BERT*) было выбрано иное направление развития — количество параметров было сокращено со 108 до 12 млн при сохранении высокой точности, что позволило использовать ее на мобильных устройствах.

Система XLM (Лампле и Конне [1348], 2019) представляет собой модель архитектуры "трансформер" с обучающими данными на нескольких языках. Такой подход полезен для машинного перевода, но также обеспечивает получение более надежных представлений для задач на одном языке. Две другие важные системы, GPT-2 (Редфорд и др. [1842], 2019) и T5 (Раффел и др. [1843], 2019), были описаны выше в этой главе. В последней статье также был представлен корпус из 35 млрд слов, получивший название *Colossal Clean Crawled Corpus* (C4).

В алгоритмах предобучения также были предложены различные многообещающие улучшения (Янг и др. [2399], 2019; Лю и др. [1431], 2019). Предобученные контекстные модели описаны Петерсом и соавт. в работе [1780] (2018), а также в статье Дай и Ле [517] (2016).

Тестовый набор GLUE (*General Language Understanding Evaluation*), включающий подборку задач и инструментов для оценки систем обработки естественного языка, был представлен Венгом и соавт. ([2293], 2018). В набор тестовых задач входят задания на предоставление ответов на вопросы, определение тональности текста, анализ семантического следования, перевод и выполнение синтаксического анализа. Модели архитектуры "трансформер" настолько доминировали в таблице лидеров (уровень показателей человека оказался в этой таблице далеко внизу, на девятом месте), что новая версия тестового набора, SuperGLUE (Венг и др. [2292], 2019), была дополнена задачами, разработанными таким образом, чтобы быть более трудными для компьютеров, но все же относительно простыми для человека.

В конце 2019 года система T5 стала абсолютным лидером с результатом 89,3, всего на пол-очка отставая от базового уровня человека, равного 89,8. По трем из десяти задач система T5 в действительности превосходит показатели производительности человека: предоставление ответа "да/нет" на вопрос (например, такой: "Находится ли Франция в том же часовом поясе, что и Великобритания?") и две задачи на понимание прочитанного, заключающиеся в предоставлении ответов на вопросы после прочтения параграфа или новостной статьи.

Машинный перевод является основной областью применения моделей языка. В 1933 году Петр Троянский получил патент на “переводящую машину”, но в то время еще не существовало компьютеров, пригодных для реализации его идеи. В 1947 году Уоррен Уивер, опираясь на работы в области криптографии и теории информации, писал Норберту Винеру: “Когда я смотрю на статью на русском языке, я говорю: «В действительности это написано на английском, но закодировано странными символами. Сейчас я приступлю к декодированию»”. Сообщество предпринимало попытки подобного декодирования, но в то время еще не было достаточного количества данных и вычислительных ресурсов, чтобы реализовать этот подход на практике.

В 1970-х годах ситуация начала изменяться и система SYSTRAN (Тома [2219], 1977) стала первой коммерчески успешной системой машинного перевода. Работа системы SYSTRAN опиралась на лексические и грамматические правила, подготовленные лингвистами вручную, а также на обучающие данные. В 1980-х годах сообщество занималось чисто статистическими моделями, основанными на частоте слов и фраз (Браун и др. [318], 1988; Коэн [1253], 2009). Но когда размер обучающих наборов достиг миллиардов и даже триллионов токенов (Брантс и др. [292], 2007), это привело к появлению систем, которые давали понятные результаты, но все же не достигающие уровня беглого языка (Оч и Ней [1704], 2004; Золлман и др. [2452], 2008). Оч и Ней в работе [1705] (2002) показывают, как дискриминативное обучение позволило достичь прогресса в машинном переводе в начале 2000-х годов.

Суцкевер и соавт. в работе [2154] (2015) впервые показали, что можно обучить машинному переводу нейронную сквозную модель класса от последовательности к последовательности. Баданау и соавт. ([109], 2015) продемонстрировали преимущества модели, которая совместно обучается выравниванию предложений в исходном и целевом языках и переводу между этими языками. Вашвани и соавт. ([2266], 2018) показали, что нейронные системы машинного перевода могут быть дополнительно улучшены путем замены сетей LSTM архитектурой “трансформер”, в которой для захвата контекста используется механизм внимания. Такие нейронные системы перевода быстро обогнали системы на базе статистических методов анализа фраз, а применение архитектуры “трансформер” очень скоро распространилось и на системы решения других задач обработки естественного языка.

Исследованиям в области предоставления **ответов на вопросы** способствовало создание SQuAD, первого крупномасштабного набора данных для обучения и тестирования систем предоставления ответов на вопросы (Раджпуркар и др. [1847], 2016). С тех пор для таких задач был разработан ряд моделей глубокого обучения (Сео и др. [2027], 2017; Кескар и др. [1219], 2019). В системе ARISTO (Кларк и др. [447], 2019) глубокое обучение используется в сочетании с ансамблем других методов. С 2018 года в большинстве моделей систем предоставления ответов на вопросы используются предобученные языковые представления, что привело к заметному улучшению их показателей по сравнению с более ранними системами.

Логический вывод на естественном языке — это задача оценки, следует ли гипотеза (*собаки должны есть*) из некоторой предпосылки (*все животные должны есть*). Этот класс задач был популяризирован появлением тестового набора PASCAL Challenge (Даган и др. [513], 2005). Сейчас для подобных систем доступны крупномасштабные наборы данных (Боумен и др. [274], 2015; Вильямс и др. [2348], 2018). Системы на основе предобученных моделей, таких как ELMO и BERT, на текущий момент демонстрируют наилучшую производительность при решении задач логического вывода на естественном языке.

Конференция *Conference on Computational Natural Language Learning* (CoNLL) фокусируется на применении обучения в области обработки естественного языка. Все конференции и журналы, упомянутые в главе 23, в настоящее время включают работы и статьи по глубокому обучению, занимающему сейчас ведущее положение в области обработки естественного языка.