

ЛЕКЦИЯ 1. Введение в трансформеры

Эволюция подходов.

За последние 20 лет произошли большие изменения в обработке естественного языка (natural language processing, NLP). Было испробовано множество разных подходов и, наконец, вступили в новую эру доминирования удивительной нейросетевой архитектуры трансформеров (Transformer). Эта архитектура глубокого обучения унаследовала многие методы своих предшественников. Ее эволюция началась с различных методов нейронного моделирования, постепенно перешла к архитектуре энкодера-декодера с механизмом внимания и продолжает развиваться. Архитектура трансформеров и ее вариации достигли успеха благодаря следующим разработкам:

- контекстно-векторное представление слов;
- улучшенные алгоритмы токенизации для обработки невидимых или редких слов. Токенизация - это процесс представления необработанного текста в меньших единицах, называемых токенами. Например, сопоставить каждое слово в тексте с числовым индексом.
- использование дополнительных токенов запоминания в предложениях, таких как Paragraph ID в Doc2vec или токен классификации CLS в языковой модели BERT;
- механизмы внимания, которые устраняют необходимость кодировать всю информацию предложения в один вектор контекста;
- механизм многопоточного самовнимания;
- позиционное кодирование порядка слов;
- параллелизируемые архитектуры, ускоряющие обучение и точную настройку;
- сжатие моделей (дистилляция, дискретизация и т. д.);
- перенос обучения (многоязычное и многозадачное обучение).

В течение многих лет мы использовали традиционные подходы NLP, такие как языковые модели n -грамм (n-gram language models), модели извлечения информации на основе TF-IDF (TF-IDF-based information retrieval models) и матрицы терминов документа с позиционным кодированием (one-hot encoded document-term matrices). Все эти подходы внесли большой вклад в решение различных задач NLP (обработка естественного языка), таких как классификация последовательностей, генерация и понимание текстов на естественном языке и т.д.

С другой стороны, у этих традиционных методов NLP (обработка естественного языка) есть свои слабые стороны, например неспособность решить проблемы разреженности (sparsity), представления невидимых слов, отслеживания протяженных зависимостей (long-term dependency) и др. Для устранения этих недостатков были разработаны подходы на основе DL (deep learning - глубокого обучения), такие как:

- RNN (recurrent neural network – рекуррентные нейронные сети);
- CNN (convolutional neural network – сверточные нейронные сети);
- FFNN (feed-forward neural network – нейронные сети с прямым распространением);
- несколько вариантов, объединяющих RNN, CNN и FFNN.

В 2013 году Word2vec – модель двухслойного кодировщика слов FFNN – решила проблему размерности, создавая короткие и плотные представления слов, которые называются векторным представлением (word embedding). Эта модель-предшественник позволяла генерировать быстрые и эффективные статические векторные представления слов. Она преобразовывала исходные текстовые данные в данные машинного обучения (точнее, самообучения) путем либо предсказания целевого слова с использованием контекста, либо предсказания соседних слов на основе скользящего окна. Создатели еще одной широко используемой и популярной модели – GloVe – утверждали, что модели, основанные на подсчете, могут работать лучше нейронных моделей. Она использует как глобальную, так и локальную статистику текстового корпуса, что-бы изучать векторы на основе статистики совпадения слов. Эта модель хорошо справляется с некоторыми синтаксическими и семантическими задачами, как показано на рис. 1.1.

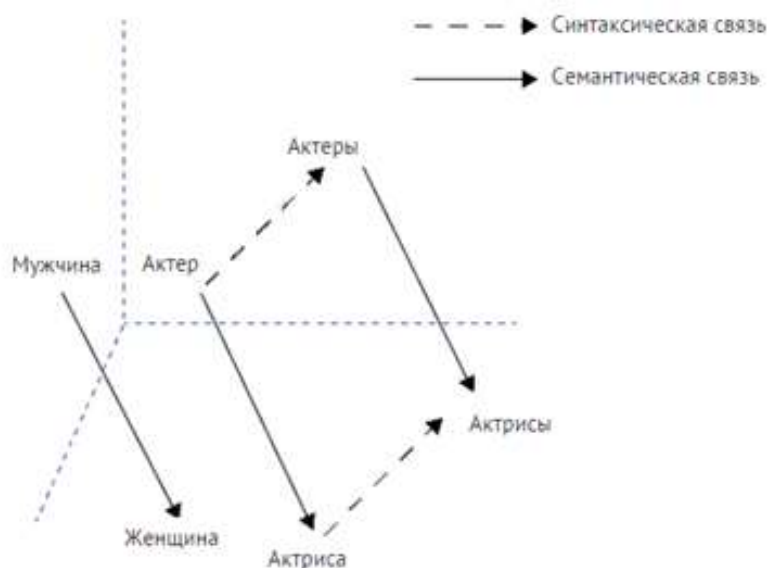


Рис. 1.1. Смещение векторов слов для извлечения отношений

На рисунке хорошо видно, что смещения векторов определений помогают сформировать векторно-ориентированное восприятие. Мы можем сформировать обобщение гендерных отношений, которое является семантическим отношением смещения между мужчиной и женщиной (мужчина женщина). Затем можем арифметически вычислить вектор термина актриса, сложив вектор термина актер и вычисленное ранее смещение. Точно так же мы можем исследовать синтаксические отношения, такие как формы множественного числа слов. Например, если известны векторы слов актер, актеры и актриса, мы можем вычислить вектор множественного числа женского рода (актрисы).

В задачах преобразования последовательности в последовательность (sequence to sequence, seq2seq) в качестве кодировщиков и декодеров стали использовать рекуррентные и сверточные архитектуры, такие как RNN, CNN и с долгой краткосрочной памятью (long short-term memory, LSTM). Основная проблема, с которой столкнулись эти ранние модели, заключалась в многозначности слов. Смысл слов приходилось игнорировать, поскольку каждому слову назначалось одно фиксированное представление, что является особенно серьезной проблемой для многозначных слов и семантики предложений.

Следующим новаторским моделям нейронных сетей, таким как универсальная языковая модель с тонкой настройкой (universal language model finetuning, ULMFit) и векторное представление на основе языковых моделей (embeddings from language models, ELMo), удалось закодировать информацию на уровне предложений и наконец разрешить (хотя и не полностью) проблему многозначности, с которой не справлялись модели со статичными представлениями слов. Эти два важных подхода основаны на сетях LSTM и реализуют концепции предварительной тренировки и тонкой настройки. Они позволяют нам применять перенос обучения, используя модели, предварительно обученные общей задаче на огромных текстовых наборах данных.

Затем мы можем легко выполнить тонкую настройку, выполнив дообучение предварительно обученной сети на размеченных данных целевой задачи. В данном случае представления слов отличаются от традиционных векторов, поскольку каждое представление слова является функцией всего входного предложения.

Современная архитектура трансформеров основана именно на этой идее.

Параллельно с эволюцией представлений слов развивалась идея механизма внимания, которая произвела сильное впечатление в области NLP и привела к значительным успехам, особенно в задачах типа seq2seq (последовательности в последовательность). Более ранние методы передавали последнее состояние

(известное как вектор контекста (context vector), или вектор смысла (thought vector), полученное из всей входной последовательности, в выходную последовательность без связывания или исключения. Механизм внимания способен построить более сложную модель, связав токены, определенные во входной последовательности, с конкретными токенами в выходной последовательности. Например, предположим, что у вас есть ключевая фраза Government of Canada (Правительство Канады) в исходном предложении, которое нужно перевести с английского на турецкий. В выходном предложении токен Kanada Hükümeti образует сильные связи с исходной фразой и более слабую связь с оставшимися словами в исходном предложении, как показано на рис. 1.2.

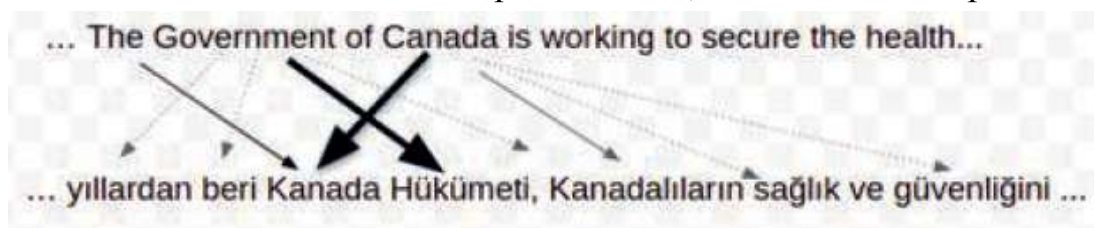


Рис. 1.2. Схематическое представление механизма внимания

Модели, использующие механизм внимания, более успешно решают задачи seq2seq, такие как перевод, ответы на вопросы и резюмирование текста.

В 2017 году была предложена успешная модель кодировщика-декодера на основе нейросети, относящейся к типу трансформеров. Ее архитектура основана на FFNN (нейронные сети с прямым распространением), но без использования рекуррентности RNN и применяет только механизмы внимания. Модели, основанные на трансформерах, к настоящему времени преодолели многие трудности, с которыми сталкивались другие подходы, и стали новой ключевой парадигмой.

Дистрибутивная семантика

Дистрибутивная семантика (Distributional semantics) описывает значение слова в виде векторного представления, в первую очередь исследуя характеристики встречаемости, а не его словарные определения. Теория предполагает, что слова, встречающиеся вместе в одной и той же среде, имеют схожие значения. Впервые ее сформулировал ученый Харрис (Distributional Structure Word, 1954). Например, слова собака и кошка чаще всего встречаются в одном и том же контексте. Одним из преимуществ дистрибутивного подхода является возможность исследовать и отслеживать так называемые лексико-семантические изменения – семантическую эволюцию слов с течением времени и в разных областях.

Традиционные подходы на протяжении многих лет опирались на языковые модели неупорядоченных наборов слов (Bag of Words, BoW) и n -граммы для построения представления слов и предложений. В подходе BoW слова и документы представляются с помощью прямого унитарного кодирования (one-hot encoding), которое является разреженным способом представления, также известным как модель векторного пространства (Vector Space Model, VSM).

Классификация текста, выявление сходства слов, извлечение семантических отношений, устранение неоднозначности смысла слов – эти и многие другие задачи NLP решали с помощью методов унитарного кодирования в течение многих лет. В свою очередь, модели языка на основе n -грамм присваивают вероятности последовательностям слов, чтобы мы могли либо вычислить вероятность того, что последовательность принадлежит корпусу, либо сгенерировать случайную последовательность на основе данного корпуса.

Метод BoW

BoW – это метод представления документов путем подсчета слов в них. Основная структура данной методики – это матрица документ–термин, представленная на рис. 1.3. Размерность матрицы очень мала (3×10), но в реалистичном сценарии размеры матрицы могут быть довольно большими, например $10 \text{ тыс.} \times 10 \text{ млн.}$

	big	cat	chased	dog	fat	mat	on	sat	slept	the
0	0.00	0.25	0.00	0.00	0.42	0.42	0.42	0.42	0.00	0.49
1	0.61	0.36	0.00	0.00	0.00	0.00	0.00	0.00	0.61	0.36
2	0.00	0.36	0.61	0.61	0.00	0.00	0.00	0.00	0.00	0.36

Рис. 1.3. Матрица терминов документа

Данная матрица основана на подсчете, где значения ячеек сформированы в соответствии с весовой схемой «частота терминат – обратная частота документа» (term frequency-inverse document frequency, TF-IDF). Этот подход не учитывает положение слов. Поскольку порядок слов во многом определяет значение фразы, его игнорирование приводит к потере смысла. Это обычная проблема метода BoW, которая наконец решена с помощью механизма рекурсии в RNN (рекуррентных нейронных сетях) и позиционного кодирования в трансформерах.

Каждый столбец в матрице обозначает вектор слова в словаре, а каждая строка обозначает вектор документа. Для вычисления сходства или несходства слов, а также документов могут применяться метрики семантического сходства. В большинстве случаев для улучшения представления документа мы используем

биграммы, такие как `cat_sat` (кот сидел) и `the_street` (улица). Могут быть построено векторное пространство, содержащее как униграммы (`big`, `cat`, `dog`), так и биграммы (`big_cat`, `big_dog`). Такие модели также называют BoW-n-граммами (`bag-of-grams`), потому что они являются естественным продолжением BoW.

Если слово обычно используется в каждом документе, как, например, английский предлог *and*, его называют часто встречающимся, или высокочастотным. И наоборот, некоторые слова почти не встречаются в документах, поэтому их называют низкочастотными (или редкими) словами. Поскольку наличие в тексте высокочастотных и низкочастотных слов может мешать правильной работе модели, в качестве решения применяют TF-IDF, который является одним из наиболее важных и хорошо известных механизмов взвешивания.

Обратная частота документа (*inverse document frequency*, IDF) – это статистический вес для измерения важности слова в документе. Например, хотя слово *the* (определенный артикль) довольно часто встречается в английском языке, у него очень маленькая различающая способность, зато слово *chased* (гналась) может быть очень информативным и давать подсказки о теме текста. Это связано с тем, что часто используемые слова (стоп-слова, функциональные слова) имеют малую различающую способность при понимании документов.

Различимость терминов также зависит от предметной области – например, список статей про глубокое обучение, скорее всего, будет содержать слово «сеть» почти в каждом документе. IDF может уменьшить веса всех терминов, используя их частоту документов (*document frequency*, DF), которая вычисляется по количеству документов, включающих термин. Частота термина (*term frequency*, TF) – это исходное количество вхождений термина (слова) в документе.

Некоторые преимущества и недостатки модели BoW на основе TF-IDF перечислены в табл. 1.

Таблица 1. Преимущества и недостатки модели TF-IDF BoW

Преимущества	Недостатки
<ul style="list-style-type: none"> • простота реализации • результаты поддаются толкованию • адаптация к предметной области 	<ul style="list-style-type: none"> • стремительный рост размерности • нет решения для невидимых слов • сложность выявления семантических отношений и синонимов • игнорируется порядок слов • медленно работает с большими словарями

Решение проблемы размерности

Для устранения размерности модели BoW широко используется латентный семантический анализ (Latent Semantic Analysis, LSA), позволяющий выявлять семантику в низкоразмерном пространстве. Это линейный метод, который фиксирует попарные корреляции между терминами. Вероятностные методы на основе LSA можно по-прежнему рассматривать как единый слой скрытых тематических переменных. Однако современные модели DL (глубокого обучения) включают в себя несколько скрытых слоев с миллиардами параметров. Кроме того, модели на основе трансформеров показали, что они могут обнаруживать скрытые представления намного лучше, чем такие традиционные модели.

Когда мы решаем задачи понимания естественного языка (natural language understanding, NLU), традиционный процесс начинается с подготовительных шагов, таких как токенизация, выделение морфологических основ (stemming), обнаружение именных словосочетаний (noun phrase detection), разбиение на части (chunking), удаление стоп-слов (stop-word elimination) и многого другого.

После этого создается матрица документ–термин на основе какого-либо алгоритма взвешивания (самым популярным остается TF-IDF). Далее эта матрица служит входными табличными данными для процедуры машинного обучения (machine learning, ML), анализа эмоциональной окраски, сходства документов, кластеризации документов или оценки степени релевантности между запросом и документом. Аналогичным образом термины, представленные в виде матрицы, можно использовать для задачи классификации токенов, включая распознавание именованных объектов, извлечение семантических отношений и т. д.

Этап классификации включает в себя прямую реализацию таких алгоритмов машинного обучения на размеченных данных, как машина опорных векторов (support vector machine, SVM), случайный лес (random forest), логистика, наивный байесовский алгоритм и множественное обучение (ускорение, или бэггинг).

Моделирование языка и генерация текстов

Традиционные подходы к решению задачи генерации текстов на естественном языке основаны на использовании языковых моделей n -грамм. Это разновидности марковского процесса (Markov process), представляющего собой стохастическую модель, в которой каждое слово (событие) зависит от подмножества предыдущих слов – униграмм (unigram), биграмм (bigram) или n -грамм (n-gram), определенных следующим образом:

- униграмма (все слова независимы и не образуют цепочки): оценивает вероятность появления слова в словаре, просто вычисляемую по его частоте вхождений в текст по отношению к общему количеству слов;

- биграмма (марковский процесс первого порядка): оценивает $P(word_i|word_{i-1})$ – вероятность $word_i$ в зависимости от $word_{i-1}$, которая просто вычисляется как отношение $P(word_i|word_{i-1})$ к $P(word_{i-1})$;
- n -грамма (марковский процесс N -го порядка): оценивает вероятность $P(word_i|word_0, \dots, word_{i-1})$.

Языковые модели могут быть реализованы с помощью библиотеки Natural Language Toolkit (NLTK). Пакет NLTK уже содержит предварительно обученную английскую модель токенизатора пунктуации для сокращенных слов и словосочетаний. Перед использованием его можно обучить пунктуации любого языка.

Использование глубокого обучения

NLP (обработка естественного языка) – одна из областей, где широко и успешно используются архитектуры глубокого обучения. На протяжении десятилетий мы были свидетелями успеха различных архитектур, особенно в области представления слов и предложений.

Векторное представление слов

Языковые модели на основе нейронных сетей эффективно решали проблемы представления признаков и языкового моделирования, поскольку стало возможным обучать более сложную нейронную архитектуру на гораздо больших наборах данных для построения коротких и плотных представлений. Разработанная в 2013 году модель Word2vec, которая сегодня является популярным методом векторного представления слов, использовала простую и эффективную архитектуру для формирования высококачественных представлений непрерывных последовательностей. Она превзошла другие модели во множестве синтаксических и семантических языковых задач, таких как анализ тональности (sentiment analysis), обнаружение перефразирования (paraphrase detection), извлечение отношений (relation extraction) и т. д. Другой ключевой фактор популярности модели – ее гораздо более низкая вычислительная сложность. Она максимизирует вероятность текущего слова с учетом всех окружающих контекстных слов или наоборот

Чтобы визуализировать векторные представления слов в трехмерном пространстве, нам нужно применить анализ главных компонент (principal component analysis, PCA). Рассмотрим пример визуализации модели с векторным представлением слов на предложениях из пьесы «Макбет» (рис. 1.4).

Взглянув на диаграмму распределения представлений, мы видим, что главные герои пьесы Шекспира – Макбет, Малкольм, Банко, Макдуф и другие – расположены близко друг к другу. Точно так же вспомогательные глаголы

английского языка shall, should и would (должен, следует и было бы) расположились рядом друг с другом в левом нижнем углу рис. 1.4.

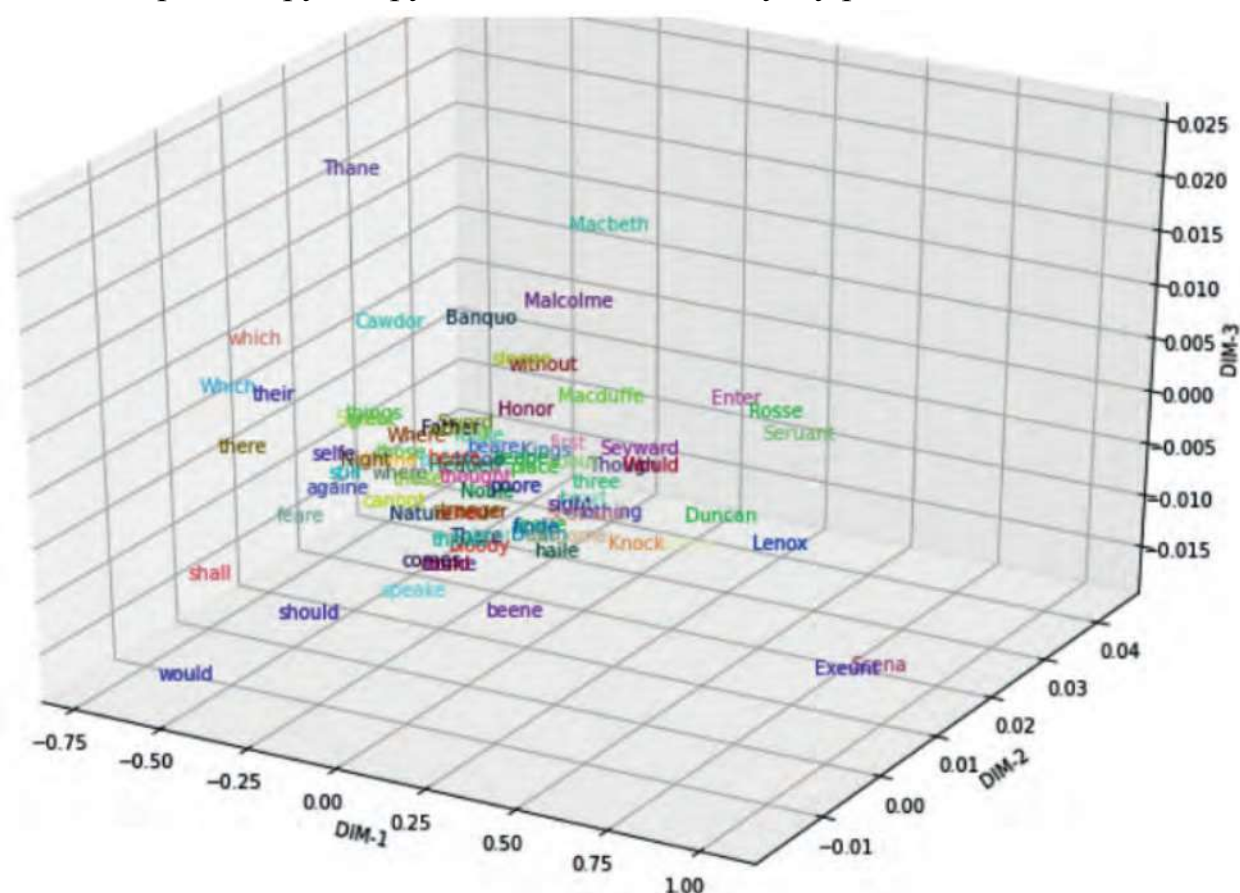


Рис. 1.4. Визуализация векторных представлений слов с помощью PCA

Используя смещение векторов, мы также можем уловить аналогию man–woman = uncle–aunt (мужчина–женщина = дядя–тетя). Более интересные визуальные примеры по этой теме можно найти в проекте по адресу <https://projector.tensorflow.org/>.

Модели, подобные Word2vec, изучают представление слов с помощью нейронной архитектуры, основанной на прогнозировании. Они используют градиентный спуск для некоторых целевых функций и предсказаний ближайших слов. В то время как традиционные подходы основаны на подсчете, архитектуры нейронных моделей основаны на прогнозировании дистрибутивной семантики. Какие методы лучше подходят для дистрибутивных представлений слов – основанные на подсчете или основанные на прогнозировании? Разработчики метода GloVe отвергают такую постановку вопроса, утверждая, что эти два подхода кардинально не отличаются. Джеффри Пеннингтон и некоторые другие исследователи даже склоняются к идее о том, что методы, основанные на подсчете, могут быть более успешными, если собирать глобальную статистику. Они заявили, что метод GloVe превзошел другие языковые модели нейронных

сетей в задачах аналогии слов, сходства слов и распознавания именованных объектов (named entity recognition, NER).

Однако эти два подхода не предлагают нам полезные решения проблем невидимых слов и неоднозначности смыслов слов. Они не используют информацию, содержащуюся в частях слов, и поэтому не могут сформировать представления редких и невидимых слов.

В еще одной широко используемой модели FastText предложен новый обогащенный подход с использованием информации, содержащейся в частях слов, где каждое слово представлено в виде набора n -грамм символов. Модель задает постоянный вектор для каждой n -граммы символа и представляет слова как суммы векторов их частей. Эту идею впервые предложил Хинрих Шютце (Hinrich Schütze; Word Space, 1993). Модель может вычислять представления даже для невидимых слов и изучать внутреннюю структуру слов, такую как суффиксы/приставки, что особенно важно для морфологически богатых языков, таких как финский, венгерский, турецкий, монгольский, корейский, японский, индонезийский и др. В настоящее время современные трансформерные архитектуры используют различные методы токенизации частей слов, такие как WordPiece, SentencePiece или Byte-Pair Encoding (BPE).

Краткий обзор RNN (рекуррентные нейронные сети)

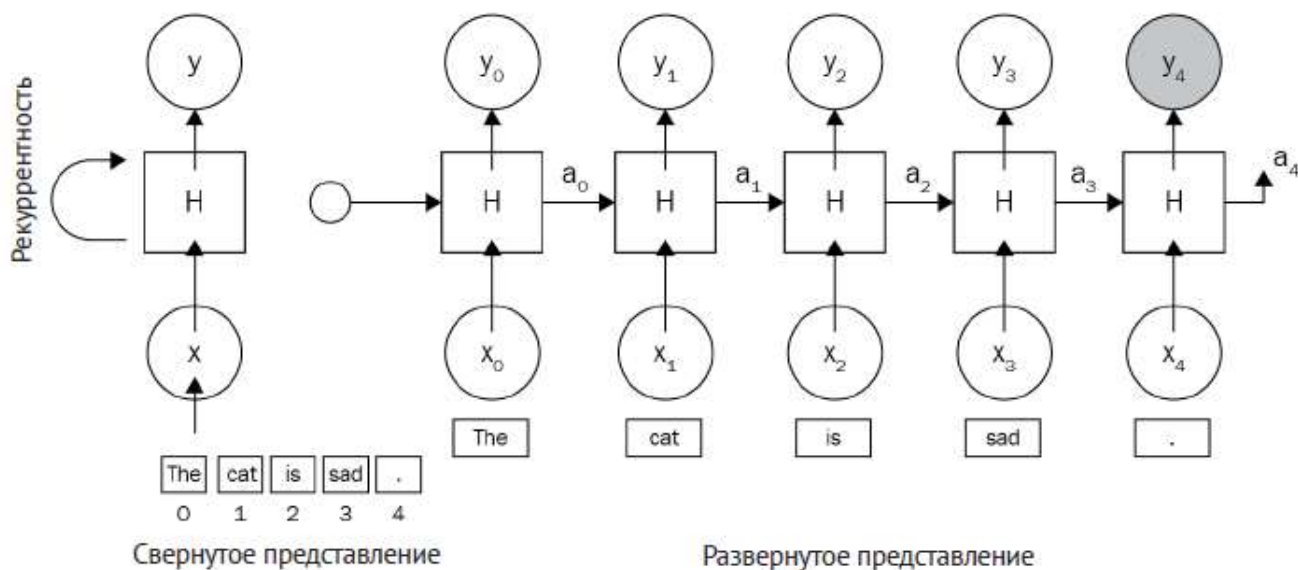
Модели на основе рекуррентной нейросети могут изучать представление каждого токена путем объединения информации о других токенах на более раннем временном шаге и изучения представления предложения на текущем временном шаге. Оказалось, что этот механизм хорошо работает во многих сценариях:

- во-первых, RNN может быть преобразована в модель «один ко многим» для генерации текстов или музыки;
- во-вторых, модели «многие к одному» можно использовать для классификации текста или анализа тональности;
- и наконец, для задач NER (Named-entity recognition – распознавание именованного объекта) используются модели «многие ко многим». Другое применение моделей «многие ко многим» – решение задач кодировщика-декодера, таких как машинный перевод, ответы на вопросы и резюмирование текста.

Как и в случае с другими нейросетевыми моделями, модели RNN принимают токены, созданные с помощью алгоритма токенизации, разбивающего весь исходный текст на элементарные единицы, также называемые токенами. Кроме того, он связывает отдельные токены с числовыми векторами –

представлениями токенов, которые изучаются во время обучения. В качестве альтернативы мы можем заблаговременно назначить задачу изучения представлений известным алгоритмам представления слов, таким как Word2vec или FastText.

Рассмотрим простой пример архитектуры RNN для предложения *The cat is sad.* (Кошка грустна), где x_0 – векторное представление слова *the*, x_1 – векторное представление слова *cat* и т. д. На рис. 1.5 показано условное (развернутое) представление RNN в виде глубокой нейронной сети (deep neural network, DNN).



Развертывание RNN (unfolding, иногда говорят раскрытие) в данном случае означает, что мы связываем с каждым словом отдельный слой нейросети. Получив предложение *The cat is sad.*, мы должны обработать последовательность из пяти слов (включая точку). Скрытое состояние на каждом слое действует как память сети. Она кодирует информацию о том, что произошло во всех предыдущих временных шагах и в текущем временном шаге, как показано на рис. 1.5.

Архитектура RNN обладает следующими преимуществами:

- входные данные переменной длины: возможность работать с вводом переменной длины независимо от размера вводимого предложения. Мы можем вводить в сеть предложения из 3 или 300 слов без изменения параметров;
- учитывает порядок слов: RNN обрабатывает последовательность слово за словом по порядку, обращая внимание на положение слов;
- подходит для работы в различных режимах (многие ко многим, один ко многим): мы можем обучить модель выполнять машинный перевод или анализировать тональность, используя общую идею рекуррентности. Обе архитектуры будут основаны на RNN.

В то же время архитектура RNN обладает и недостатками:

- проблема отдаленной зависимости: когда мы обрабатываем очень длинный документ и пытаемся связать термины, которые находятся далеко друг от друга, нам нужно принять во внимание и закодировать все другие нерелевантные термины между этими терминами;
- склонность к проблемам с разрывающимся или исчезающим градиентом: при работе с длинными документами обновление весов самых первых слов является большой проблемой из-за исчезающего градиента, что делает модель необучаемой;
- трудно применить параллельное обучение: распараллеливание разбивает основную проблему на более мелкие и выполняет решения одновременно, но RNN основана на классическом последовательном подходе. Состояние каждого слоя сильно зависит от предыдущих слоев, что исключает распараллеливание;
- медленные вычисления из-за большой длины последовательности: RNN может быть очень эффективной при работе с коротким текстом. Но более длинные документы она обрабатывает очень медленно (вспомним о проблеме отдаленной зависимости).

Хотя RNN теоретически может извлекать информацию из этапов, задолго предшествующих текущему, в реальном мире такие проблемы, как длинные документы и проблема отдаленной зависимости, не позволяют в полной мере воспользоваться этой возможностью.

LSTM (долгая краткосрочная память) и управляемые рекуррентные блоки

Модели LSTM и GRU (gated recurrent unit – управляемые рекуррентные блоки) – это новые варианты RNN, в которых решена проблема отдаленной зависимости. В частности, различные варианты модели LSTM были специально разработаны для решения проблемы отдаленной зависимости. Преимущество модели LSTM заключается в том, что она использует дополнительное состояние ячейки, которое показано в виде горизонтальной линии в верхней части блока LSTM на рис. 1.6. Это состояние ячейки контролируется специальными гейтами (gate), которые позволяют выполнять операции забывания, вставки или обновления.

Мы можем принимать следующие решения:

- какую информацию мы будем хранить в состоянии ячейки;
- какая информация будет забыта (удалена).

Чтобы узнать состояние I токенов традиционной RNN, мы вынуждены периодически обрабатывать все состояния предыдущих токенов между $timestep_0$

и $timestep_{i-1}$. Перенос всей информации из более ранних временных шагов порождает проблему исчезающего градиента, что делает модель необучаемой. Механизм гейтов в LSTM позволяет архитектуре пропускать некоторые несвязанные токены на определенном временном шаге или запоминать состояния отдаленных этапов, чтобы узнать текущее состояние токена.

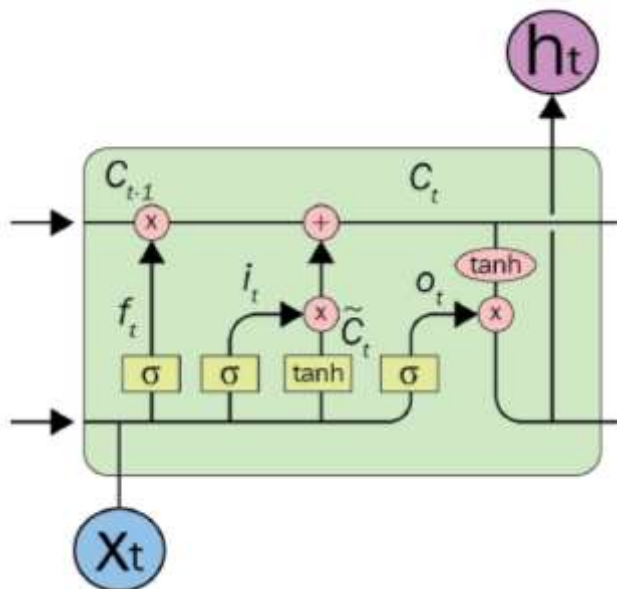


Рис. 1.6. Ячейка LSTM

Архитектура модели GRU во многом похожа на LSTM; главное отличие состоит в том, что GRU не использует состояние ячейки. Ее архитектура упрощена за счет переноса функциональности состояния ячейки в скрытое состояние, и она содержит только два гейта – гейт обновления и гейт сброса. Гейт обновления определяет, сколько информации из предыдущего и текущего временных шагов будет продвинуто вперед. Эта функция помогает модели сохранять актуальную информацию из прошлого, что также сводит к минимуму риск исчезновения градиента. Гейт сброса обнаруживает нерелевантные данные и заставляет модель их забыть.

Краткий обзор CNN (сверточные нейронные сети)

После того как CNN хорошо зарекомендовали себя в области компьютерного зрения, их начали применять в NLP для моделирования предложений и решения таких задач, как семантическая классификация текста. CNN состоит из сверточных слоев, за которыми во многих случаях следует плотная нейронная сеть. Сверточный слой работает с данными, извлекая полезные признаки. Как и в любой модели глубокого обучения, сверточный слой фактически предназначен для автоматизации извлечения признаков. В случае NLP на этот слой признаков поступают данные от слоя представления, принимающего в качестве входных данных предложения в векторизованном

унитарном формате. Унитарные векторы генерируются по token-id для каждого слова, образующего предложение. На рис. 1.8 показано представление предложения I saw a cat. (Я видел кошку) в виде унитарных векторов слов.

	1	2	3	4	5
I	1	0	0	0	0
saw	0	1	0	0	0
a	0	0	1	0	0
cat	0	0	0	1	0
.	0	0	0	0	1

Рис. 1.8. Представление предложения в виде унитарных векторов

Каждый токен, представленный унитарным вектором, подается на слой представления (embedding layer). Слой представления может быть инициализирован случайными значениями или с помощью предварительно обученных векторов слов, таких как GloVe, Word2vec или FastText. Затем предложение будет преобразовано в плотную матрицу с размерностью $N \times E$ (где N – количество токенов в предложении, а E – размер представления). На рис. 1.9 показано, как одномерная CNN обрабатывает эту плотную матрицу.

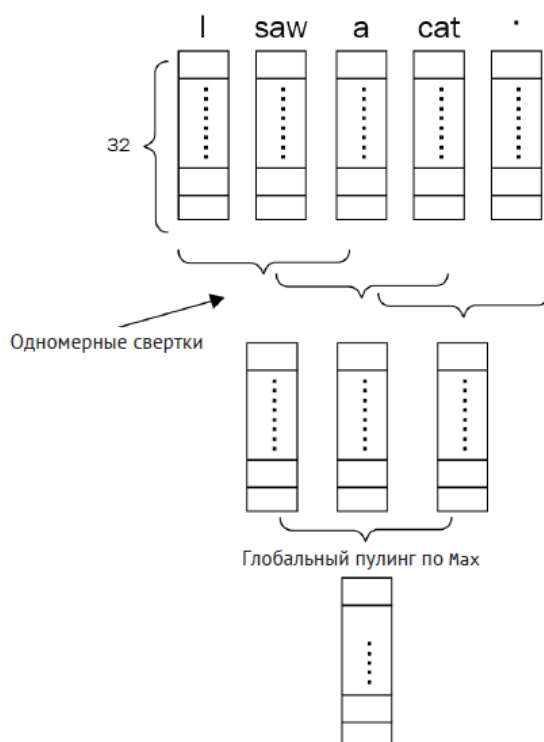


Рис. 1.9. Обработка предложения из пяти токенов в одномерной CNN

Свертка будет самой первой операцией, выполняемой с разными слоями и ядрами. Гиперпараметры сверточного слоя – это размер ядра и количество ядер. Также полезно отметить, что здесь применяется одномерная свертка, и причина этого в том, что представление токенов можно рассматривать только целиком, и

мы хотим применить ядра, способные одновременно видеть несколько токенов в последовательном порядке. Вы можете считать это чем-то вроде n -граммы с заданным окном. Один из интересных вариантов – использование неглубоких TL (transfer learning – перенос обучения) в сочетании с моделями CNN. Как показано на рис. 1.10, мы также можем проходить сети с помощью комбинации нескольких представлений токенов.

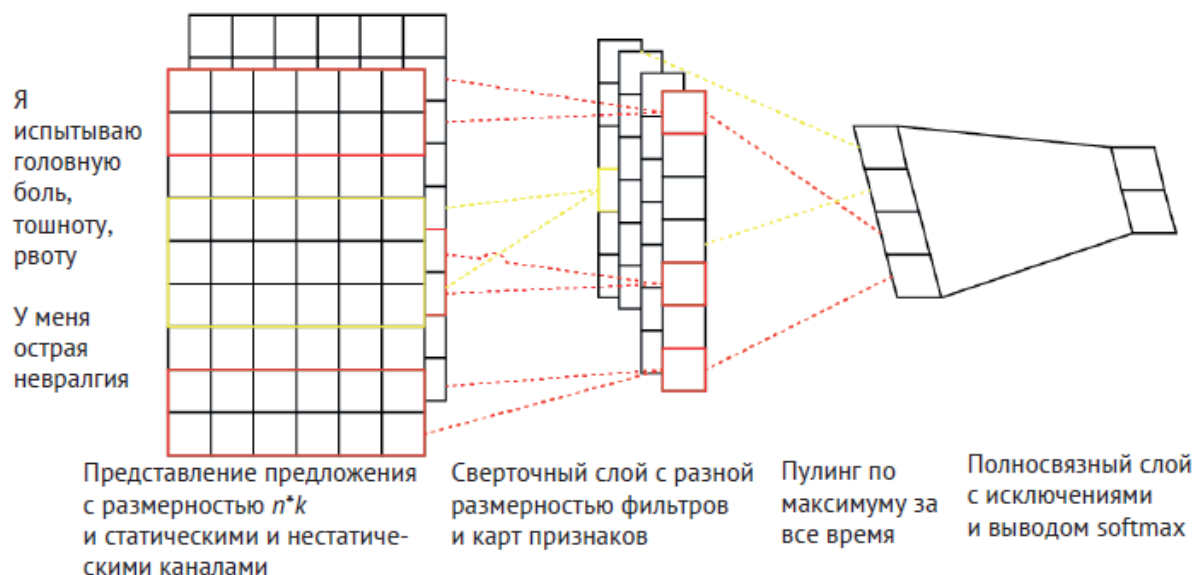


Рис. 1.10. Комбинация нескольких представлений в CNN

Например, мы можем использовать три слоя представления вместо одного и объединить их для каждого токена. При такой схеме определенный токен, например чувствую, будет представлен вектором с размерностью 3×128 , если размерность всех трех различных представлений равна 128. Эти векторные представления могут быть инициализированы предварительно обученными векторами из Word2vec, GloVe и FastText. Операция свертки на каждом шаге будет видеть N слов с соответствующими тремя векторами (N – размер фильтра свертки). Тип свертки, который используется здесь, – это одномерная свертка. Размерность – количество направлений перемещения при выполнении операции. Например, двумерная свертка будет перемещаться по двум осям, а одномерная свертка – только по одной оси. На рис. 1.11 показаны различия между ними.




Размерность свертки	Вход	Фильтр	Выход
1-мерная 	3-мерный	3-мерный	2-мерный
2-мерная 	4-мерный	4-мерный	3-мерный
3-мерная 	5-мерный	5-мерный	4-мерный

Рис. 1.11. Направления перемещения свертки

Оказывается, модель CNN демонстрирует производительность, сопоставимую со своим аналогом LSTM. Хотя CNN стали стандартом в обработке изображений, мы видели много успешных применений CNN в области NLP. В то время как модель LSTM обучена распознавать закономерности, развернутые во времени, модель CNN распознает закономерности в пространстве.

обработке естественного языка (natural language processing, NLP)

Далее мы обсудим классические архитектуры глубокого обучения (deep learning, DL), такие как рекуррентные нейронные сети (recurrent neural network,

RNN), нейронные сети с прямым распространением (feed-forward neural network, FFNN) и сверточные нейронные сети (convolutional neural network, CNN).