Способы уменьшения вычислительной сложности нейросетевых языковых моделей

В. Я. Чучупал, в.н.с.,
Вычислительный центр им. А. А. Дородницына ФИЦ ИУ РАН

Аннотация

Нейросетевые языковые модели существенно превосходят традиционные статистические модели по достигаемым показателям качества и начали широко использоваться в коммерческих системах распознавания речи. Вместе с тем им присущи заметные недостатки, одним из которых является очень большая вычислительная сложность. В статье дан краткий обзор основных современных подходов понижения вычислительной сложности нейросетевых языковых моделей. Рассмотрены методы компенсации влияния размера словарей, повышения быстродействия за счет учета свойств графических ускорителей. Приведены результаты оптимизации вычислительной сложности для моделей внимания и трансформера, связанные с использованием методов передачи знаний, динамической оптимизации размеров моделей, свойств функции внимания, а также оптимизации разрядности представления информации в нейросетях. Показано, что комбинирование таких подходов и приёмов позволяет на порядки ускорить процессы обучения и распознавания, симметрично снизив требования к объему памяти.

Ключевые слова: автоматическое распознавание речи, нейро-сетевые модели языка, вычислительная сложность моделей языка, иерархический софт-макс, трансформер, модель внимания.

Статистические модели языка

Создание и развитие моделей нейросетевых ЯМ привело к качественному улучшению характеристик моделей языка: величина перплексии на основных тестовых корпусах данных снизилась по сравнению с общепринятыми граммными статистическими моделями в несколько раз, до значений, которые недавно представлялись маловероятными. Важное преимущество нейросетевых ЯМ перед статистическими заключается в использовании действительных векторных представлений слов с мерами сходства, которые коррелируют с их семантическими

и синтаксическими признаками. Другое важное преимущество состоит в использовании на порядки более длительных по времени контекстов из слов.

В рамках существующего экстенсивного подхода к развитию нейросетевых моделей качество современных ЯМ существенно зависит от их архитектуры и от размеров и свойств обучающих текстовых корпусов. Экстра-класса нейросетевые ЯМ [1] построены на сверх-глубоких (десятки слоев) архитектурах, так, что могут иметь сотни миллиардов параметров и использовать огромные выборки данных [2] для обучения. Соответственно, эти модели обладают очень высокими требованиями к вычислительным ресурсам (доступным далеко не всем разработчикам).

Поэтому стала актуальной задача разработки эффективных не только по показателям качества, но и в вычислительном плане моделей языка.

Модель языка определяет вероятность появления в речи заданного набора слов:

$$P(W) = P(w_1, \ldots, w_N) = \prod_{n=1}^{N} P(w_n | w_1 \ldots, w_{n-1}) \quad (1)$$

Обычно эта вероятность определяется через условные вероятности появления слов при заданном предыдущем контексте, то есть если w — текущее слово, а h — предыдущие слова, то модель языка определяет вероятности $p(w \mid h)$ для всех пар (h,w).

Максимальная длина контекста n в (1) ограничивается некоторым значением m. Таким образом вероятность $P(W)$ вычисляется из вероятностей -грамм:

$$P(W) = P(w_1, \ldots, w_N) \approx \prod_{n=1}^{N} P(w_n | w_{n-m} \ldots, w_{n-1}) \quad (2)$$

Аппроксимация в виде произведения вероятностей n-грамм (2), обрезанных контекстов, очевидно является неточной, особенно при небольших значениях n.

Для нейросетевых языковых моделей число элементов последнего слоя сети обычно равно размеру словаря, так, что вероятность любого слова $w \in V$ в контексте H вычисляется как нормализованная (с помощью софт-макс) величина активации $f(w)$ соответствующего этому слову элемента слоя:

$$P(w) = \frac{e^{f(w)}}{\sum_{w' \in V} e^{f(w')}} = \frac{e^{f(w)}}{Z} \quad (3)$$

Сумма экспонент активаций всех элементов слоя (слов словаря) — нормирующий член Z в знаменателе (3) называется статистической суммой. При больших размерах словаря вычисление Z вычислительно трудоемко. Далее, при использовании логправдоподобия как функции потерь и градиентных методов обучения, градиент от логарифма вероятности появления слова (3) по весам сети [3,4] равен:

$$\frac{\partial P(w,\theta)}{\partial \theta} = \frac{\partial f(w,\theta)}{\partial \theta} - \sum_{w' \in V} P(w') \frac{\partial f(w',\theta)}{\partial \theta} \quad (4)$$

что можно интерпретировать как вычитание из величины градиента активации элемента, который соответствует корректному слову, среднего (матожидания) значений градиентов активаций всех элементов слоя. То есть в (4) также используются вероятности всех элементов слоя, что вычислительно трудоёмко.

Несмотря на разнообразие архитектур нейросетей, грубая оценка их вычислительной сложности в принципе несложна, так как выполняемый набор операций ограничен. В то же время за счет использования расширенных наборов инструкций процессоров и графических ускорителей вычисления можно проводить параллельно. Векторные и матричные операции в графических ускорителях в некотором диапазоне размерностей выполняются практически за то одно и то же время. Учет наличия или отсутствия всех этих факторов загромождает реальные оценки сложности вычислений, тем более с учетом наличия или отсутствия векторных операций в системе команд процессоров. Обычно операции сложения, умножения рассматриваются как эквивалентные, более того, выполняемые одновременно.

Пусть на вход слоя размером n элементов подается последовательность из l признаковых векторов размерности d. Для направленной сети (персептрон) пусть входом являются склеенные k таких векторов. Тогда оценка вычислительной сложности одного слоя: направленной сети будет порядка O(l · k · n · d) операций, рекуррентной сети : O(l · d2), сверточной сети с шагом s, длиной ядра n, размерности вектора признаков d и числом ядер k: $O\left(\frac{l}{s}\times n\times d^2\times k\right)$. При оценке общего количества вычислений нужно учесть, что объем вычислений растет пропорционально числу слоев.

Эти оценки с практической точки зрения слишком грубы, поскольку не учитывают возможность параллельных вычислений. Все операции внутри слоев сети можно выполнять параллельно, т.е. сложность слоя — константа O(1). Вычисления в направленных и сверточных слоях также не зависят от результатов в предыдущие моменты времени. Но для вычисления активаций скрытого слоя рекуррентной сети в любой момент времени нужно знать активации в предыдущий момент. Даже при распараллеливании сложность слоя будет расти линейно со временем — O(l).

Специфичным для нейросетевых языковых моделей для слов является ёмкость и вычислительная сложность последнего слоя сети, на котором вычисляется распределение вероятностей слов. Для словаря из нескольких миллионов слов и размерности предыдущего слоя порядка тысяч элементов, последний слой будет содержать миллиарды оцениваемых параметров. Нормирование элементов этого слоя будет включать оценку миллионов экспонент.

Другая особенность связана с вычислительной сложностью функции внимания, используемой в моделях кодера-декодера и трансформера. Функция внимания использует сразу весь входной сигнал, поэтому её сложность пропорциональна длине входа, а с учётом того, что она,

в свою очередь, применяется на каждом входном элементе, итоговая сложность — квадратичная O(l^2), то есть её сложно реально использовать на больших (например, длиной 32K) входных последовательностях.

Эта обстоятельства объясняют то факт, что алгоритмы вычислений нейро-сетевых ЯМ нуждаются в оптимизации с целью сокращения вычислительных требований.

Понижение вычислительной сложности путем оптимизации словарей

Использование слов как единиц языкового моделирования вполне естественно. Словные языковые модели обеспечивают более высокую (с использованием существенно меньших по длине контекстов) точность распознавания речи по сравнению с моделями на подсловных (буквы, кусочки слов — морфы) единицах. Вместе с тем упомянутая выше исключительная вычислительная сложность последнего слоя фактически не дает возможность в полной мере использовать словари из миллионов слов.

Известны несколько методов аппроксимации процедуры использования полного словаря.

Селекция слов словаря с учетом их важности

Простым и реально применяемым подходом является сокращение размера словаря до приемлемого размера при имеющихся вычислительных ресурсах. Например, в словаре оставляют наиболее частотные слова, а для остальных используют специальное обозначение, символ для незнакомых (несловарных) слов. Либо последовательно удаляются слова, менее всего влияющие на корпусную перплексию. Таким образом словарь приводится к требуемому размеру.

Очевидный недостаток такого подхода связан с тем, что результаты распознавания на таком сокращённом словаре будут содержать много появлений символа незнакомого слова.

Языковые модели для подсловных единиц

Более эффективным подходом к проблеме компенсации влияния размеров словаря является использование словарей из подсловных единиц, в качестве которых выбираются кусочки слов (морфы) или буквы. Общее количество морфов контролируемо и обычно выбирается относительно небольшое (до десятков тысяч) и тогда объем памяти и вычислений на последнем слое сети не является существенным.

Бонусом применения словарей из подсловных единиц является то, что при этом попутно снимается проблема появления незнакомых слов: любое произнесение будет представлено буквами.

Разработаны простые и эффективные вычислительные алгоритмы, которые позволяют рассчитать оптимальные, заданного размера, словари из морфов, по обучающим текстам и критериям оптимальности словарей.

Наиболее часто используется метод попарного байтового кодирования BPE (byte-pair encoding). Словарь вычисляется так, чтобы для записи текста требовалось минимальное количество байтов (длина кодов морфов – константа).

Принципиально алгоритм BPE имеет следующий вид:

Алгоритм 1 Метод попарного побайтового кодирования BPE

1. Инициализировать словарь морфов алфавитом из букв (начальных и в середине слов), синтаксических символов.
2. Вычислить частоты встречаемости для всех биграмм из морфов словаря на текстовом корпусе.
3. Найти биграмму из п.2. с наивысшей частотой встречаемости
4. Создать новый морф из биграммы и занести его в словарь.
5. Если размер словаря морфов превысил порог, перейти на п.8.
6. Вычислить частоты биграмм из морфов с учетом нового морфа.
7. Перейти на п.3.
8. Стоп.

Словари из морфов широко используются в современных передовых нейро-языковых моделях, фактически став общепринятым подходом к построению словарей в больших нейро-ЯМ.

Обычно используются словари размером более 20 тысяч морфов. В этом случае фактически вычисляется гибридный словарь, который включает в себя кроме кусочков слов также и целые слова.

Недостатком подсловных ЯМ является то, что морфы несут меньше семантико-синтаксической информации чем слова. Из-за размера единиц языковые модели для морфов требуют существенного увеличения длины используемого контекста по сравнению с контекстом из слов. Языковые модели на уровне слов в большей степени используют синтаксические и семантические закономерности, поэтому при прочих равных условиях обеспечивают меньший уровень ошибок распознавания. Увеличение длины контекста при использовании морфов также сказывается на увеличении вычислительной сложности и более сложном обучении.

Факторизация словарей. Иерархический софт-макс

При обучении языковых моделей с использованием максимально правдоподобных оценок обычно не требуется вычислить вероятности появления всех слов. Нужна вероятность корректного слова. В режиме распознавания похожая ситуация: используются вероятности появления небольшого числа наиболее правдоподобных в данный момент слов. Поэтому стандартная процедура вычисления и нормализации вероятностей всех слов словаря избыточна.

Можно разбить словарь на непересекающиеся кластеры слов и вычислять вероятности слов через вероятности этих кластеров и вероятности слов внутри содержащих их кластеров. Поскольку по таком подходе требуется знать только вероятности появления кластера и слова внутри него, вычислительная сложность будет существенно снижена.

Одним из самых известных алгоритмов такого рода является иерархический софт-макс [5] (Hierarchical SoftMax, HSM), в котором линейное представление словаря заменяется на древовидное. Вершины дерева соответствуют кластерам из слов, слова — листьям. Вычисление вероятности слова сводится к вычислению произведения вероятностей вершин дерева, лежащих на пути от корня к листу слова.

Оценка вероятности некоторой n-граммы $p(w\_t \mid w_{\_1},\ldots,w_{\_t\text{-}n}) = p(w_t \mid h_t)$ для словаря в виде двухуровневого дерева, вершины которого соответствуют кластерам слов, имеет вид [6]:

$$p(w_t|h_t) = p(w_t|C(w_t), h_t) \times p(C(w_t)|h_t) \quad (5)$$

где $C(w_t)$ — класс слова w_t.

Каждая из перемножаемых вероятностей $p(w_t \mid C(w_t), h_t)$ и $p(C(w_t) \mid h_t)$ вычисляется с помощью обычной софт-макс языковой модели, но со словарем из слов, которые содержатся в соответствующей вершине дерева. Допустим она v содержит слов, $v << |V|$, тогда сложность вычислений падает с $O(|V|)$ (при отсутствии древовидного представления) до $O(v)$ операций вычисления экспонент. В случае оптимального бинарного разбиения словаря по частотному принципу можно обойтись вычислением в среднем $\log_2(V)$ вероятностей в вершинах дерева.

Способ конструирования кластеров может быть различен, формально он не важен: любое разбиение будет обеспечивать корректность (5). Можно, например, использовать кластеры из семантически или синтаксически близких слов или составленные по принципу частоты встречаемости.

Важным является то, что метод HSM может использоваться как при обучении моделей, так и при распознавании.

Недостатком этого метода является потенциальная потеря качества модели. В каждой вершине дерева находится своя языковая модель, которая вносит погрешность в оценку вероятностного распределения. Произведение таких аппроксимаций (5) имеет ошибку большего порядка. Это приводит к заметному увеличению перплексии модели по сравнению с языковой моделью с полным софт-максом (3), то есть для всех слов ЯМ [5].

Оптимизация путем использования свойств графических ускорителей и многоядерности процессоров

Алгоритм HSM, вообще говоря, не определяет однозначно принцип разбиения слов на кластеры. Тем не менее очевидно, что выбор классов слов может иметь большое значение.

Известен целый ряд модификаций метода HSM [7] за счет стратегии выбора кластеров. Метод адаптивного HSM [8] построен, как и другие на использовании частотных свойств слов. Кроме этого, он существенно использует свойства программного обеспечения и графических ускорителей, GPU, которые используются для реализации нейромоделей языка.

Слова упорядочиваются по частотности и для уменьшения количества ветвлений дерева кластер начальной вершины содержит не только список кластеров следующего уровня, но и набор самых частотных слов (краткий список, short-list). Таким образом для оценки вероятности часто используемых слов не нужно вычислять произведение (5). На основе экспериментальной зависимости времени вычислений от размеров кластеров для выбора оптимального состава (количества и размеров) кластеров используется принцип динамического программирования.

В целом такая стратегия при исследованных типах GPU (NVIDIA M40, K40) показала, что оптимальное число кластеров при двухуровневом дереве (т.е. корень-вершина-лист) лежит в диапазоне от 10 до 15, однако фактически субоптимальные результаты в [8] получены уже при использовании только двух кластеров (списка из наиболее частотных слов и кластера-хвоста из остальных), таким образом факторизация (5) используется не всегда. Возможно этим и объяснятся то, что в адаптивном HSM удается получить оценки перплексии, не уступающие полному методу с нормировкой вероятностей всех слов словаря.

Понижение вычислительной сложности путем сэмплирования словаря

Альтернативный подход к проблеме компенсации влияния размеров словаря используют методы, основанные на сэмплировании слов. Идея заключается в том, чтобы при вычислении оценок (3) и (4) вместо активаций всех элементов (слов словаря) использовать только их небольшую часть. Одним из первых был предложен [3] метод выбора по важности (Importance Sampling, IS). Если для оценки суммы Z использовать часть слов, то, чтобы результат был ближе к истинному, нужно выбирать самые вероятные слова. Вероятности слов $P(w) = \frac{e^{S(w)}}{z}$ неизвестны, поэтому используют априорное распределение, равномерное или степень униграммного, в соответствии с которым выбирается заданное количество слов, которые используется для вычисления (3), (4).

Для того, чтобы учесть в оценках Z отличие априорного распределения от истинного, вероятности сэмплированных слов можно скорректировать весами.

Среднее некоторой функции $g(y)$ на множестве $y \in Y$ определяется как $\Sigma_{y \in Y} P(y) g(y)$

Аппроксимация среднего $g(Y)$ по точкам y_i, $i = 1,...N$, $y\_(i) \sim P(y)$ имеет вид:

$$\overline{g(Y)} = \frac{1}{N} \sum_i P(y_i) g(y_i) \qquad (6)$$

Чтобы аналогичная аппроксимация по точкам, выбранным для иного распределения $y\_i \sim Q(y)$ давала такое же значение, как в (6), нужно скорректировать вероятности $Q(y_i)$ точек весами $P(y_i)/Q(y_i)$. При этом неизвестные вероятности $P(y_i)$ точек y_i аппроксимируют в соответствии с (3), повторно используя значения y_i, $i = 1,...N$.

Метод IS вычислительно эффективен и дает достаточно хорошие результаты (тем не менее также как и метод HSM он уступает по достигаемым значениям перплексии оценкам (3) на полном словаре). Качество метода IS возможно страдает из-за того, что аппроксимацию вероятностей с использованием выбранных слов приходится делать дважды: при оценке статистической суммы Z и для коррекции вероятностного распределения.

Важным ограничением метода является то, что он существенно сокращает вычислительные затраты только при обучении моделей, но не при распознавании.

Повторной аппроксимации вероятностей используя выбранные слова можно избежать, если изначально искать нормированные распределения. В конкурентном методе оценки по контрасту с шумом, NCE (Noise Contrastive Estimation)[9] сэмплирование слов словаря используется, чтобы построить альтернативное распределение при решении задачи оценки параметров бинарного классификатора, который различает правильное слово и слова шумного распределения (выбранные слова моделируют шум).

В этом случае оптимизируемый критерий — это отношение вероятностей, причем статистическая сумма фигурирует как в числителе, так и в знаменателе. Оценки вероятности фактически будут близки к нормированным (значение статистической суммы в данном случае оцениваемый параметр алгоритма, но экспериментально показано, что выбор =1 не влияет на результаты).

Метод NCE позволяет получить несколько меньшую корпусную перплексию, чем метод ds,jhf по важности, IS. Однако, как и IS, метод NCE заметно сокращает вычислительные затраты только при обучении моделей.

Использование вычислительно эффективной архитектуры сети

Как уже было отмечено, конструкция рекуррентных сетей удачно подходит для моделирования временных рядов, в частности языковых последовательностей, однако при использовании многослойных сетей с большим числом параметров и обучающих данных очевидным недостатком рекуррентных сетей становится сложность распараллеливания операций. Более эффективными с этой точки зрения являются конструкции на основе направленных, в частности, свёрточных сетей, которые изначально разрабатывались для обработки временных рядов.

В режиме параллельных вычислений свёрточные и направленные сети имеют константную сложность $O(1)$. Фактически полного распараллеливания операций достичь обычно не удается, особенно в сетях большой сложности, тем не менее модели, основанные на относительно простых (вариант сети TDNN) свёрточных сетях, например Wave2Letter [10] показали преимущество в скорости вычислений в режиме распознавания выше на 1–2 порядка по сравнению с аналогами на рекуррентных сетях.

Вычислительно эффективная модель трансформера.

Лучшие по характеристикам современные нейросетевые языковые модели являются вариантами трансформера. Сам трансформер также можно рассматривать как

существенно переработанную, глубокую модификацию модели LAS [11], в которой рекуррентные слои и слой внимания заменены на свёрточные слои самовнимания и внимания.

Использование модели внимания позволило получить выдающиеся результаты. Но они связаны с ростом вычислительных затрат. Алгоритмы внимания используют весь распознаваемый сигнал, от начала, до конца. В этом случае вычислительная сложность растёт пропорционально квадрату длины входного сигнала $O(l^2)$, то есть превышают вычислительную сложность рекуррентной сети $O(l)$.

Если учесть, что продвинутые модели имеют очень большое число слоёв (до сотен), то даже при использовании мощных многопроцессорных и многоядерных конфигураций обучение моделей может занять недели, так как вычислительные затраты достигают тысяч петафлопс/секунда-дней [1]. В режиме распознавания условие предоставления слоям внимания всего сигнала означает задержку начала выдачи результатов распознавания, как минимум, до окончания речи. Вместе с высокими вычислительными затратами это означает, что трансформер не может работать в режиме реального времени.

По этой причине исследование вычислительно экономичных модификаций трансформера является одним из наиболее востребованных направлений.

По этим причинам системы распознавания речи на основе трансформера обычно используют языковые модели на основе морфов, таким образом избегая дополнительных вычислений, связанных с большими словарями и необходимости использования иерархического софт-макса или методов, использующих сэмплирование слов словаря.

Другим перспективным направлением уменьшения объема вычислений и размеров используемой памяти является использование разряженности значений внимания: после вычисления значений софт-макс значимые величины, как правило, сосредоточены в небольшом числе точек, остальными можно пренебречь без существенных потерь в точности.

На основе экспериментов с глубиной слоев сети и выбором оптимального числа её блоков в [12] предложен подход к построению трансформера с существенно меньшим числом параметров, что в данном случае пропорционально снижает и объем вычислений. При 2.8 раз меньшем числе параметров, чем стандартный трансформер, этот вариант модели обеспечивает такие-же значения показателей качества. При некотором увеличении числа параметров (все ещё в 1.8 раза меньше, чем у стандартной модели) показатели качества превышают значения для стандартной модели.

Подход основан на изменении архитектуры блоков трансформера: использовании в начале блоков слоёв переменной размерности и глубины: на входе сети располагаются блоки с более узкими и мелкими слоями,

соответственно блоки у выхода содержат более широкие и глубокие слои. Эта модель трансформера в целом в 2–4 раза глубже, чем стандартная.

Использован только один фокус (head) внимания. Вместо много-фокусного внимания на входе блоков сети использованы слои переменной (в зависимости от места блока) глубины, которые реализуют групповые преобразования, проецируя входные признаки или активации в пространства большей размерности и затем возвращая или редуцируя их в пространства исходной размерности. Использование дополнительных слоёв для групповых преобразований в составе блоков приводит к увеличению общей глубины нейросети, в экспериментах число слоев сети достигало 222.

Другая модель трансформера, «быстроформер», полученная в результате детальной многоаспектной оптимизации с целью повышения скорости работы сети во время распознавания предложена в [13]. Оптимизация вычислительной нагрузки осуществлялась несколькими путями: использованием метода дистилляции знаний для уменьшения размеров сети, структурированной обрезкой параметров сети, квантованием параметров модели и оптимизацией вычислений в реальном времени за счет разработки и использования многопоточного и многозадачного программного обеспечения.

Дистилляция знаний заключается в том, что очень большая модель, часто это универсальная модель типа BERT (bidirectional encoder representations from transformers) [14], используется как учитель для обучения меньшей по размерам модели (ученик). В данном случае оказалось, что модель — ученик могла без потери качества содержать в 3 раза меньше слоёв с размером в 2.5 раза меньше, чем модель-учителя, что пропорционально снижало требуемые вычислительные ресурсы.

Процедуры обрезки или удаления элементов нейросети основаны на измерении величины важности её параметров и элементов, например, по наблюдаемым значениям градиентов функции потерь на контрольных данных, с последующим удалением наименее важных. Обычная процедура обрезки заметно уменьшает количество параметров, но не гарантирует увеличения вычислительной эффективности при использовании матричных операций и GPU, поскольку удаляются достаточно мелкие единицы, а количество и размерность крупных единиц, таких как фокусы и слои сети, может не меняться. В случае структурной обрезки её объектами были фокусы внимания и слои поточечных направленных сетей, удаление которых приводило к изменению размерностей матриц и давало вычислительный выигрыш при использовании многоядерных процессоров. Например, ускорение в примерно 3 раза при незначительной потере точности достигалось при обрезке половины фокусов внимания и трёх четвертей промежуточных, скрытых направленных поточечных слоёв.

Квантование параметров моделей позволяло понизить разрядность значений весов и активаций до 16-бит для реализации в графическом ускорителе или 8-бит для центрального процессора. Такое понижение точности как уже известно [13] не влияет практически на результат работы трансформера. В то же время использование новых инструкций типа 8-ми битных AVX (Advanced Vector eXtension)-512-VNNI (Vector Neural Network Instructions) для процессоров Intel Cascade Lake CPU может существенно ускорить работу нейросети. Точно также графические ускорители NVIDIA с ядром Volta поддерживают эффективные 16-битные вычисления.

В данном случае использование матричных операций с квантованными значениями позволило ускорить вычисления в трансформере в 3.5 раза по сравнению с использованием исходных значений.

Применение описанных выше приёмов вычислительной оптимизации трансформера привело к тому, что построенная в результате нейросеть на тестовых задачах распознавала в десятки (в одной задаче более чем в двести) раз быстрее, что стандартная модель. В то же время процедура реализация даже части упомянутых выше подходов представляется нетривиальной, трудоёмкой и не гарантирующей эффективности полученного решения в каждом конкретном случае.

Оптимизация была направлена на увеличение быстродействия и более экономное использование памяти именно в режиме распознавания в то же время процедура обучения из-за использования методов дистилляции знаний, структурной обрезки и квантования в результате стала существенно сложнее.

В этом плане дополняющая, эффективная по памяти и в первую очередь в режиме обучения, модификация модели трансформера может быть получена за счет пересмотра алгоритмов вычислений в модели самовнимания [15]. В частности, при использовании совпадающих запросов и ключей (часто применяется и очевидно имеет физический смысл) свойство разряженности значений внимания позволяет понизить вычислительную сложность путем применении т.н. локально чувствительного хэширования (LSH, locality sensitive hashing), когда с большой вероятностью близкие по расстоянию (L_2) вектора кодируются одним хэшем, а вектора, удаленные друг от друга — другим. В этом случае элементы слоя внимания распадаются на классы хэшей и вычисления софт-макса для запроса можно проводить только для ключей его хэша.

При обучении многослойных сетей с использованием градиентных методов (обратного распространения ошибки) помимо величин весов нейронов нужно также хранить величины их активаций, на что в многослойной сети уходит много памяти — пропорционально числу слоев и размеру каждого. В [15] структура блока трансформера изменена таким образом, что он реализует т.н. обратимую остаточную сеть [16] (reversible resisual network, RevNet), в которой активации элементов слоя могут быть вычислены из активаций элементов последующего слоя, то есть хранить их в памяти не нужно. Кроме того, поскольку в однонаправленных позиционно-независимых слоях над каждым входным вектором выполняются идентичные и независимые операции, их выполнение можно реорганизовать: вместо одной матричной операции большой размерности выполнить несколько аналогичных меньшей размерности, что требует пропорционально меньшей памяти во время вычислений, хотя с точки зрения времени вычислений, при использовании GPU, оно, скорее всего, увеличится.

Применение описанных выше алгоритмов и усовершенствований, по оценкам самих авторов [15] привело к существенному понижению сложности

вычислений и объему требуемой памяти, например, для внимания оценка составила порядка вместо и даже. С практической точки зрения это дало возможность использовать модель трансформера на не супер-мощных конфигурациях (8 GPU или 8 TPU) для работы (обучение языковых моделей и машинный перевод) с последовательностями до 64K символов длиной.

Заключение

Самые высокие показатели качества языковых моделей достигаются сейчас при использовании нейросетевых моделей. Программные реализации методов, полученные прямым кодированием алгоритмов, как правило, имеет слишком большую вычислительную сложность для того, чтобы их можно было использовать в практических системах. Одним из источников вычислительной сложности, в частности, является размер словаря систем распознавания. При использовании языковых моделей для слов как единиц языка, размеры словарей достигают миллионов слов. Разработан целый рад субоптимальных методов, которые существенно снижают вычислительную загрузку вплоть до того, что позволяют обучать модели с использованием достаточно больших датасетов на относительно недорогих, массового класса, вычислительных конфигурациях. Появившиеся недавно языковые модели на основе трансформера практически сразу же позволили получить передовые результаты на большинстве популярных индикативных тестов на качество обработки, распознавания и смысловой интерпретации языка. Однако вычислительная сложность таких моделей настолько велика, что они фактически используются в небольшом количестве коллективов, которые имеют доступ к соответствующим вычислительным мощностям. По этой причине одним из новых и наиболее востребованных направлений исследований является разработка вычислительно экономных модификаций моделей трансформера и лежащей в его основе модели внимания. Работы в этом направлении только начались, но уже предложенные методы позволяют существенно понизить вычислительную сложность модели, правда ценой существенного усложнения и специализации алгоритмов под определенные классы задач.

Список литературы

1. Language Models are Few-Shot Learners / T. B. Brown [и др.]. — 2020. — arXiv: 2005.14165 [cs.CL].
2. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer / C. Raffel [и др.]. — 2019. — arXiv: 1910.10683 [cs.LG].
3. *Yoshua Bengio J.*-S. S. Quick Training of Probabilistic Neural Nets by Importance Sampling // Mathematiques Centre. — 2002. — Дек.– №2.
4. *Labeau M., Allauzen A.* Learning with Noise-Contrastive Estimation: Easing training by learning to scale. // Proceedings of the 27th International Conference on Computational Linguistics. — Santa Fe, New Mexico, USA : Association for Computational Linguistics, 2018. — С. 3090–3101. — URL: https://www.aclweb.org/anthology/C181261.
5. A Neural Probabilistic Language Model / Y. Bengio [и др.] // J. Mach. Learn. Res. — 2003. — Т. 3. — С. 1137–1155. — ISSN 1532–4435.
6. Recurrent neural network based language model / T. Mikolov [и др.] // Interspeech. — 2010.
7. *Chen Welin G. D., Michael A.* Strategies for training large vocabulary neural language models. — 2015. — arXiv: arXiv:1512.04906[cs.CL].

8. Efficient softmax approximation for GPUs / E. Grave [и др.]. — 2016. — arXiv: arXiv:1609.04309[cs.CL].
9. *Gutmann M., Hyvarinen A.* Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. // AISTATS: Thirteenth International Conference on Artifcial Intelligence and Statistics, PMLR 9. 2010. — №. 297–304.
10. Wav2letter++: The Fastest Open-source Speech Recognition System / V. Pratap [и др.]. — 2018. — arXiv: arXiv:1812.07625v1[cs.CL]. — URL: www.arxiv.org/pdf/1812.07625.pdf 990.
11. *Chan W., Jaitly N., Quoc V. Le O. V.* Listen, Attend and Spell. — 2015. — arXiv: arXiv:1508.01211v2[cs.CL].
12. DeLighT: Very Deep and Light-weight Transformer / S. Mehta [и др.]. -2020. — arXiv: arXiv:2008.00623v1[cs.LG] — URL: www.arxiv.org/pdf/2008.00623.pdf.
13. *Kim Y. J., Awadalla H. H.* Highly Efficient Transformer Models for Natural Language Understanding. — 2020. — arXiv: arXiv:2010.13382v1[cs.CL] — URL: www.arxiv.org/pdf/2010.13382.pdf.
14. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding / J. Devlin [и др]. — 2019. – arXiv: arXiv:1810.04805v2[cs.CL] — URL: www.arxiv.org/pdf/1810.04805v2.pdf.
15. *Kitaev N., Kaiser L., Levskaya A.* Reformer: The E_cient Transformer. — 2020. — arXiv: arXiv:2001.04451v2[cs.LG] — URL: www.arxiv.org/pdf/2001.04451v2.pdf.
16. The reversible residual network: Backpropagation without storing activations. /A. N. Gomez [и др.] // Advances in neural information processing systems. Т. 30 / под ред. I.Guyon [и др.]. — Curran Associates, Inc., 2017. –С. 2214_2224. – URL:https://proceedings.neurips.cc/paper/2017/file/f9be311e65d81a9ad8150a60844bb94c-Paper.pdf.

WAYS TO REDUCE THE COMPUTATIONAL COMPLEXITY OF NEURAL LANGUAGE MODELS

V. Y. Chuchupal,
Candidate of Physical and Mathematical Sciences, Leading Researcher at the A. A. Dorodnitsyn Computing Center of the Federal Research Center «Informatics and Management» of the Russian Academy of Sciences.
E-mail: v.chuchupal@gmail.com

Abstract

Neural language models are significantly superior to statistical models and have begun to be widely used in commercial speech recognition systems. At the same time, they have noticeable disadvantages, one of which is their great computational complexity. The article provides a brief overview of the main modern approaches to reducing the computational complexity of neural network language models. Methods to compensate for the growth of computations due to the size of dictionaries, increasing the speed using the properties of the properties of graphics accelerators are considered. The results of computational complexity optimization for the attention and transformer models, based on the use of knowledge transfer, dynamic control of the size of models, the use of attention properties, as well as a optimization of information representation depth. It is shown that the combination of such approaches and techniques makes it possible to accelerate the learning and interference processes by orders of magnitude, as well as symmetrically reduce the required memory volumes.

Keywords: automatic speech recognition, neural language models, complexity of neural models, hierarchial softmax, transformer, attention model.