Содержание

Предисловие

В современном мире голосовых помощников и умных вещей непреодолимое желание человека автоматизировать любую часть рутинного процесса находит для распознавания речи все больше и больше применений. Такой большой интерес к развитию речевых технологий поспособствовал появлению массы научных работ по данной тематике. А с развитием интернета и вычислительных технологий человечество научилось хранить и обрабатывать объемы данных, немыслимые по меркам пятидесятилетней давности.

Эти и другие факторы сыграли ключевую роль в сдвиге направления новейших исследований в области распознавания речи, которая изначально считалась ответвлением цифровой обработки сигналов, в сторону нейронных сетей и end-to-end подходов в частности. Все большее распространение, особенно в LVCSR (large vocabulary continuous speech recognition), получают end-to-end подходы на основе Connectionist Temporal Classification (CTC) loss, encoder-decoder модели (LAS: Listen, Attend and Spell), в том числе онлайновые, Recurrent Neural Network Transducer (RNN-T) и другие. End-to-end модели позволяют значительно упростить традиционный пайплайн обучения ASR из-за своей простоты и в то же самое время добиться значительных выигрышей в качестве за счёт большой обобщающей способности.

В данной работе рассматриваются методы дискриминационного обучения end-to-end акустических моделей, и предлагается новый подход на основе сэмплирования из списка N лучших гипотез и дальнейшего сэмплирования путей из этих гипотез. Для достижения этого результата имплементируется свой онлайновый динамический декодер на C++ и быстрый подсчёт forward-backward переменных для сэмплирования. Производится сравнение с классическим CTC бейзлайном и наблюдаются улучшения в целевой метрике от 5% до 15% в зависимости от датасета. Предложенный алгоритм для дискриминационного обучения работает со скоростью, сравнимой с классическим CTC loss и не требует построения решёток гипотез или хранения графа декодирования в памяти.

1 Введение

1.1 История распознавания речи

Задача распознавания речи (automatic speech recognition, ASR) издавна волновала умы ученых. Сложность задачи состоит в том, что помимо задачи классификации, где каждому фрейму аудио необходимо поставить в соответствие тот или иной класс (фрагмент транскрипции), модель распознавания речи должна во время тренировки выучить *выравнивания* (alignments) между аудио и транскрипцией, так как они обычно не доступны в обучащией выборке. Если говорить в терминах математической статистики, то случайные величины "аудио"и "транскрипция"являются наблюдаемыми, а выравнивания (какой именно кусок аудио соответствует какой части транскрипции) являются скрытыми переменными. Для того, чтобы успешно распознавать речь модель должна не только во время инференса выводить, какой именно класс представлен на данном кусочке аудио, но и во время обучения понимать, какой именно класс является истинным для каждого момента времени в звуке.

Первые системы по распознаванию произнесенных цифр (spoken digit recognition, что по сути является более простой версией задачи по распознаванию речи) стали появляться в начале 1950-ых.

В 1952 году в недрах Bell Laboratories была построена электрическая схема для распознавания аудиозаписей изолированно произнесенных цифр от нуля до девяти [1]. В данной системе предполагалось, что на вход поступает полное произношение одной цифры. Это сильно упрощало задачу по сравнению с ASR, так как при такой постановке не требовалось находить границы отдельно произнесенных слов в рамках целой фразы, которые [слова] в спонтанной речи, могут быть произнесены без пауз, либо с небольшими паузами. Переформулируя в терминах выравниваний: не требовалось находить скрытые переменные, и вся задача сводилась к классификации целого звука на один из 10 клаассов.

Статистические подходы, которые позволяли моделировать акустические свойства фразы целиком, в том числе и скрытые перемнные, стали появляться в 1980-ых, когда в научном сообществе получили широ-

кое распространение скрытые марковские модели (Hidden Markov Model, HMM) [2] [3]. Такой подход к решению задачи по автоматическому распознаванию речи встраивал скрытые переменные, соответствующие выравниванию между произнесенной фразой и ее транскрипцией в саму статистическую модель HMM, с помощью которой потом осуществлялся статистический вывод алгоритмом Baum-Welch для нахождения скрытых параметров. Baum Welch испольует внутри себя простую технику динамического программирования для нахождения forward переменных $\alpha_i(t)$ и backward переменных $\beta_i(t)$. В качестве условных вероятностей наблюдений (фрагментов аудио) при условии состояния в HMM использовалась смесь гауссовских распределений (GMM), обычно с диагональной матрицей ковариаций. Состояниями скрытой марковской цепи могли быть контекстно-независимые фонемы (CI phones, monophones), контекстно-зависимые тройки фонем (CD phones, triphones), кластеры из контекстно-зависимых фонем (сеноны). Улучшениями классических GMM-HMM можно также считать speaker adaptive training (SAT), а также различные дискриминативные критерии: MMI, bMMI, MPE и другие. Хороший обзор классических статистических методов можно найти в [4].

В 2000-ых и 2010-ых широкое распространение получил гибридный подход к акустическому моделированию, который сочитал классический подход и передовые исследования в нейросетях для предсказания апостериорных вероятностей фонем (трифонов, сенонов) при условии входной фразы [5]. Исследовалось как применения обычных feed-forward сетей с левым и правым контекстом [6] [7], так и рекуррентных сетей с контекстом из всей фразы [8]. Гибридные системы обучались поверх уже существующих классических, так как для обучения методом обратного распространения ошибки с кроссэнтропийной функцией потерь требовалась информация о выравниваниях между аудио фреймами и сенонами. Для построения выравниваний обычно использовались лучше GMM-HMM системы, например CD-GMM-HMM + SAT + bMMI.

Параллельно с этим развивается end-to-end подход к распознаванию речи, в котором нейросеть тренируется напрямую на исходных данных (аудио файлах и транскрипциях), без промежуточного шага в виде гене-

рации выравниваний. При этом способе решения задачи нейросеть неявно обучается находить выравнивания сама, и хотя, отход от априорных знаний, используемых в классических гибридных системах, может сказаться положительно на обучающей способности, для полноценной тренировки такой системы требуется количество данных и вычислительные мощности, недоступные еще несколько лет тому назад. Основную техническую проблему здесь составляет то, что длина выхода (транскрипции) на порядки отличается от длины входа (аудио признаков). Задача предсказания произвольной выходной последовательности по входной в общем случае называется sequence-to-sequence [9] и встречается также в том или ином виде в областях машинного перевода (Neural Machine Translation, NMT), распознавания жестов (gesture recognition) и рукописного текста (handwriting recognition).

Существует несколько подходов к решению задачи sequence-to-sequence. В Neural Machine Translation, например, применяются encoder-decoder архитектуры [10] с attention механизмом. Encoder Decoder архитектура может также применятся и в распознавании речи, например, как это делается в статье Listen, Attend and Spell [11]. Однако без применения дополнительных трюков LAS не является онлайновым распознаванием речи, т.е. он не может обрабатывать речь в потоке. Эту проблему призван решить RNN Transducer [12], но ценой того, что RNN-T отсутствует явный attention механизм, которым может управлять декодер. Вместо этого маргинализация по всем выравниваниям происходит в самой функции потерь. Аналогичная маргинализация по всем выравниваниям, правда без участия декодера происходит и в задачах по распознаванию речи и рукописного текста, где длина выходной последовательности оказывается меньше длины входной и применяется также connectionist temporal classification loss (CTC loss) [13] [14] [15]. Строго говоря, CTC loss не является полностью end-to-end подходом, в том смысле что во время inference для достижения хороших результатов ему всё равно требуется принимать beam search декодирование с внешней языковой моделью (двум другим end-to-end подходам для этого достаточно beam search decoding без внешней LM и second pass rescoring с внешней LM [16]). Но

он является end-to-end в том смысле, что не требует знания о выравниваниях и о лексиконе (фонемном словаре). Классический пайплайн распознавания речи требует лексикон, так как он необходим для обучения качественных GMM-HMM бейзлайнов для генерации выравниваний.

1.2 Feature extraction

В данном разделе будет описана природа звука и в какой-то степени описан процесс, как именно человек воспринимает звуковые колебания и выделяет из них речь. Как известно из школьного курса физики, звук представляет собой упругие продольные колебания в некоторой среде. Характеристики среды напрямую влияют на параметры звука – скорость распространений, силу затухания, максимальную амплитуду, частоту звука и другие.

Математически звук представляет собой некоторый сигнал, отображающей временную ось $\mathbb{R}_+$ в некоторый ограниченный отрезок интенсивностей $I \subset \mathbb{R}$. На практике часто берут $I = [-1, 1]$. Таким образом определена функция $f : \mathbb{R}_+ \mapsto [-1, 1]$, график которой называется вейвформой. Для цифровой обработки сигналов звук дискретизуется как в пространственном домене, так и в апмлитудном. Частота, с которой записываются значения ампилтуды, называется частотой дисретизации D, а величина, равная двоичном логарифму от количества возможных значений ампилтуды – bit depth. Таким образом $\hat{f} : A \mapsto B$, $A = \{\alpha \cdot i \mid i \in \mathbb{N}_0\}, B = \{\beta \cdot i \mid i \in \mathbb{Z}\}, \alpha = 1/D, \hat{f}(x) = [f(x)]$.

На практике $\hat{f}$ представляет собой вектор размерности $\approx 10^5$, с которым не удобно работать статистическими методами машинного обучения (например, мешает проклятие размерности). Помимо этого, числа в векторе слабо интерпретируемы – одни фонемы перетекают в другие и жёсткой сегментации речи (кроме пауз) на акустические единицы в вейвформах не наблюдается.

В связи с этим вейвформы часто подвергаются дальнейшней обработке. Классическое волновое уравнения записывается следующим образом:

$$\Delta u(\vec{x}, t) = \frac{1}{c^2} \cdot \frac{\partial^2 u(\vec{x}, t)}{\partial t^2}$$

Следующее параметрическое семейство функций является его решением:

$$f^{(1)}_{A,\vec{a},\omega,\varphi_0}(\vec{x}, t) = A \sin((\vec{a}, \vec{x}) + \omega t + \varphi_0)$$
$$f^{(2)}_{A,\vec{a},\omega,\varphi_0}(\vec{x}, t) = A \cos((\vec{a}, \vec{x}) + \omega t + \varphi_0)$$

$c^2 = \frac{\omega^2}{(\vec{a},\vec{a})}$ – квадрат скорости распространения волны в среде. Кроме того, любая линейная комбинация этих функций также является решением. Фурье изучал уравнения теплопроводности, которые по форме сходны с волновыми уравнениями и пришёл к выводу, что при соблюдении некоторые условий любое решение дифференциального уравнения можно представить, как сумму (возможно, бесконечную) базисных функций – гармонических колебаний, описанных выше.

В связи с этим вместо самой вейвформы разумно использовать её спектр. Полный обзор пайплайна по извлечению фичей для распознавания речи приведён в [17] [18], здесь же мы приведём краткий обзор этого пайплайна.

Для начала сигнал разбивается на фреймы – короткие промежутки с пересечением (обычно промежутки составляют в длину 25 мс, с шагом 10 мс). В рамках каждого фрейма предполагается, что звук стационарен, значит спектр – модуль коэффициентов преобразования Фурье от фрейма – интерпретируем, и значение i-того коэффициента равно амплитуде той части сигнала, частота которой равна γi Hz для некоторой фиксированной γ, зависящей от размера окна и частоты дискретизации. На фрейм из K отсчётов получается K коэффициентов Фурье, первый коэффициент – это коэффициент перед константной базисной функцией, он задаёт то, что в литературе называют DC offset – среднее значение по фрейму. DC offset всегда действительный, если преобразование Фурье посчитано от действительного сигнала. Следующие $\lfloor \frac{K}{2} \rfloor$ задают амплитуды и фазы соответствующей базисной функции ($e^{-i\frac{2\pi k}{N}}$) в форме $A \cdot e^{i\phi_0}$. Остальная половина равна отражённой и комплексно сопряжённой первой половине. В спектр для фрейма попадают DC offset и первая половина коэффициентов, предварительно прошедших через комплексный модуль. Получившиеся вектора из $F := \lfloor \frac{K}{2} \rfloor + 1$ действительных элементов на каждый фрейм формируют прямоугольную матрицу размера $T \times F$, которая и называется спектрограммой.

Известно, что восприятие громкости и частоты человеческим ухом нелинейно. Например, на основе этого построена система музыкальных нот: чатота ноты в двух соседних октавах различается ровно в 2 раза.

Аналогично, шкала громкости в децибелах также является нелинейной: Изменение в амплитуде в 2 раза соответствует линейному приращению на ≈ 6 дБ. В связи с этим коэффициенты спетрограммы проходят через дополнительные преобразования. Шкала частот в герцах переводится в шкалу частот в мелах по следующей формуле:

$$m = 2595 \cdot \log_{10}\left(1 + \frac{f}{700}\right) = 1127 \ln\left(1 + \frac{f}{700}\right)$$

Поскольку мы работаем с дискретными векторами вместо функций, для такого перевода необходимо осуществлять интерполяцию. Обычно берут линейную сетку в пространстве мелов, переводят её в герцы, и в каждом узле размещают треугольный фильтр, который скалярным произведением умножается на весь спектр. Получившие значения называют мел фильтрбанками. Далее, вместо линейных значений амплитуды используют логарифмические, что примерно соответствует шкале в децибелах, поэтому mel filterbanks логарифмируются для получения logmel filterbanks.

Logmel filerbanks уже можно передавать на вход модели, в этом работе рассматривается именно такой урезанный пайплайн, но стоит упомянуть про дополнительные шаги, которые иногда делают для классических ASR систем. Было замечено, что фильрбанки также обладают некоторой периодичностью, поэтому для каждого фрейма от logmel коэффициентов берут дискретное косинусное преобразование (DCT, discrete cosine transform), вторую половину коэффициентов преобразования также выкидывают (т.к. она симметрична первой половине). Получившиеся коэффициенты называются MFCC (mel frequence cepstrum coefficients). Cepstrum – перестановка букв в слове spectrum. Кроме самих коэффициентов также конкатенируют их поэлеметную разность (delta) и разность разностей (delta-delta). Получившийся вектор фичей (обычно 40 фичей для 13 MFCC) подают в акустическую модель.

1.3 Акустические модели

Рассмотрим последовательность фреймов аудио $x_1, \dots, x_T \in \mathbb{R}^k$. Пусть $y = y_1 \dots y_N \in \mathcal{L}^N$ – искомая транскрипция аудио. Акустической моделью называется такая статистическая модель, которая связывает распределение коэффициентов фичей аудиофрейма с буквой / фонемой, которая была произнесена на данном фрейме. Т.к. $N \neq T$ (обычно $N << T$), априорно мы незнаем, какому фрейму соответствует какой лейбл. Это информацию даёт выравнивание – вектор с элментами из $\mathcal{L}$ длины T. Если задан один такой вектор, говорят о жёстком выравнивании (hard alignment), если распределение на векторах – то о мягком выравнивании (soft alignments). Обычно жёсткое выравнивание является модой распределения мягких выравниваний: $z^* = \arg\max_z P(z|x, y)$. Стоит понимать, что при переходе от soft alignments к hard alignments теряется часть информации. Например, на шумном аудио может быть невозможно однозначно сказать, к какому фрейму аудио принадлежит та или иная буква, но мы можем обладать списком наиболее вероятных гипотез, т.е. распределение $P(z|x, y)$ может быть мультимодальным. В случае с hard alignments нам из этого списка придётся выбрать какое-то одно из выравниваний. Перейдём к различным способам акустического моделирования.

1.3.1 Gaussian mixture models

GMM (Gaussian Mixture Model) – одна из самых простых статистических моделей, которая может связать распределение компонент вектора фичей аудиофрейма с тем, какая именно фонема представлена на этом фрейме. Предположим, что у нас имеется база аудио с правильными выравниваниями, т.е. каждый фрейм аудио мы имеем какой-то лейбл. GMM предполагает, что распределение фрейма x при условии лейбла a, $P(x|a)$, является смесью гауссовских распределений, т.е.

$$P(x|a) = \sum_{i=1}^{M} \frac{c_{i,a}}{(2\pi)^{k/2}|\Sigma_{i,a}|^{k/2}} \exp\Big(-\frac{1}{2}(x - \mu_{i,a})^T \Sigma_{i,a}^{-1}(x - \mu_{i,a})\Big)$$

Набор параметров распределений: $\Theta_a = \{c_{i,a}, \mu_{i,a}, \Sigma_{i,a}\}$ можно найти с помощью MLE, но это сложно, т.к. функция правдоподопия не выпуклая, и для подсчёта градиента правдоподобия необходимо дифференцировать логарифм суммы экспонент, что может вызвать проблемы с численной стабильностью. Вместо этого для нахождения параметров смеси гауссовских распределений используют Expectation-Maximization (EM) алгоритм. Кратко EM-алгоритм можно описать следующим образом: вводяться скрытая переменная Z, которая принимает одно из M значений и описывает принадлежность аудиофрейма какому-то определённому кластеру из смеси распределений. Далее повторяется следующие 2 шага до сходимости: Expectation и Maximization. На E-шаге рассчитываются апостерирорные вероятности принадлежности каждого элемента датасета каждому из элементов смеси, с учётом параметров $\Theta_a^{(i)}$ на данном шаге. На M-шаге апостериорные вероятности принадлежности считаются фиксированными, и находится оценка максимального правдоподобия для $\Theta_a^{(i+1)}$ при условии фиксированных вероятностей принадлежности.

Часто в GMM матрицы ковариаций считают диагональными $\Sigma = diag(\sigma_1^2, \sigma_2^2, ..., \sigma_k^2)$. Может показаться, что это уменьшает ёмкость моделей, но это не совсем так: из-за наличия многих компонент в смеси. Смесь гауссовских распределений с диагональными матрицами ковариаций может моделировать при достаточно большом количестве компонент одно распределение с полной матрицей ковариаций.

Смесь гауссиан хорошо приближает распределение векторов акустических фичей [19], если из них убрать временную информацию, т.е. рассматривать векторы акустических фичей как совокупность, а не как последовательность. Например, GMM используется как Universal Background Model в задаче speaker verification. Гауссовское распределение хорошо приближает многие реальные физические процессы из-за закона больших чисел, а их смесь позволяет прближать мультимодальные распределения. Кроме того, MLE оценки гауссовского распределения хорошо интерпретируемы и просты для вычислений. Из недостатков GMM можно отметить невозможность их использования для описания динамики

случайного процесса, где каждые следующие выпадения случайных величин зависят от предыдущих. Кроме того, чтобы GMM успешно описали распределение, находящееся вдоль малоразмерного многообразия в пространстве большой размерности, необходимо очень много параметров. Например, чтобы описать двумерную сферу S^2 в $\mathbb{R}^n$ необходимо "покрыть"гауссовскими распределениями всю её поврехность, в то время как для описания S^{n-1} в $\mathbb{R}^n$ необходимо одно гауссовское распределение (оно будет "полым"из-за проклятия размерностей). Широко известно [20], что распределение акустических фичей речи как раз является малоразмерным многообразием в пространстве большой размерности, т.к. сгенерировано физической системой с большим количеством ограничений и небольшим количеством степеней свободы.

1.3.2 Hidden markov models

Как уже отмечалось выше, гауссовские смеси не подходят для моделирования зависимостей в последовательностях переменной длинны. Одной из первых статистических моделей для моделирования временной зависимости между акустическими векторами и элементами выравнивания в распознавании речи стала модель скрытой марковской цепи.

Рассмотрим $y_0, y_1, y_2, ... \in S$ – случайные переменные, образующие марковскую цепь. S – конечное пространство состояний. $\rho_0(y_0)$ – распределение в начальный момент времени. Марковость цепи означает, что выполняется марковское свойство:

$$P(y_n|y_{n-1}, y_{n-2}, \dots, y_0) = P(y_n|y_{n-1})$$

То есть условная вероятность выпадания следующего состояния зависит только от предыдущего состояния. Так как состояний конечное число, то веротяность переходов можно описать матрицей $A = (a_{ij})_{i,j \in S}$: $a_{ij} = P(y_n = i|y_{n-1} = j)$. Таким образом марковская цепь – это тройка (S, ρ_0, A).

В распознавании речи марковские цепи используются для описания транскрипций. S – это в простейшем случае пространство букв (фонем). Для семплирования из марковской цепи необходимо засемплировать из

ρ_0 начаально состояние, а потом с помощью матрицы A семплировать шаг за шагом следующие состояния. Вероятность нахождения в момент времени t в состоянии i также легко находиться:

$$P(y_t = i) = \rho_t(i) = (A^t \rho_0)_i$$

Скрытая марковская модель устроена следующим образом: помимо марковской цепи и её случайных величин y_i, которые теперь называются скртыми переменными, вводяться дополнительные случайные величины – наблюдения $x_i \in \mathbb{R}^k$. Каждой скрытой перемнной в выборке из скрытой марковской модели соответствет одно наблюдение. Для описания наблюдаемых параметров при условии скрытых используется ещё одно распредлеение:

$$P(x_t | y_t = i) = b_i(x_t)$$

.

Выборка из скрытой марковской цепи делается следующим образом: изначально из начального распределения ρ_0 на скрытых параметрах выбирается первое скрытое состояние $y_0 \sim \rho_0$. Далее, из его распределения для наблюдений выбирается первое наблюдение: $x_0 \sim b_{y_1}$. Далее, $y_1 \sim a_{*y_0}, x_1 \sim b_{y_1}, y_2 \sim a_{*y_1}$ и так далее. Получается последовательность наблюдений $x_0, x_1, x_2 \ldots$ и соответствующая им последовательность скрытых состояний $y_0, y_1, y_2 \ldots$.

В распознавании речи наблюдения – это акустические вектора признаков, последовательность скрытых состояний – выравнивание. В качестве модели для распределения наблюдений при условии скрытого состояния $b_i(x)$ используется смесь гауссиан.

Для обучения HMM-GMM, т.е. нахождения оптимальных параметров Θ гауссовских смесей и вероятностей переходов HMM используют вариацию EM алгоритма, называемую алгоритмом Баума-Велша. Одна итерация алгоритма состоит, как обычно, из expectation и maximization шагов. Для выполнения expectation шага нам необходимо знать forward переменные α и backward переменные β, определённые следующим образом:

$$\alpha_i(t) = P(x_0, x_1, \ldots, x_t, y_t = i|\Theta)$$

– вроятность вывести последовательность наблюдений $x_0, \ldots, x_t$ и оказаться в состоянии i.

$$\beta_i(t) = P(x_{t+1}, \ldots, x_T|y_t = i, \Theta)$$

– вероятность находясь в состоянии i вывести последовательность наблюдений $x_{t+1}, \ldots, x_T$.

Обновляются forward и backward перемнные с помощью динамического программирования и следующих тождеств:

$$\alpha_i(0) = \rho_0(i)b_i(x_0)$$

$$\alpha_i(t+1) = b_i(x_{t+1}) \sum_{j=1}^{N} a_{ij}\alpha_j(t)$$

$$\beta_i(T) = 1$$

$$\beta_i(t) = \sum_{j=1}^{N} \beta_j(t+1)a_{ij}b_j(x_{t+1})$$

Зная forward и backward переменные, можно посчитать следующие значения:

$$\gamma_i(t) = P(y_t = i|X, \Theta) = \frac{\alpha_i(t)\beta_i(t)}{\sum_{j=1}^{N} \alpha_j(t)\beta_j(t)}$$

– апостериорная вероятность нахождения в состоянии i в момент времени t при условии наблюдаемой последовательности.

$$\xi_{ij}(t) = P(y_t = i, y_{t+1} = j|X, \Theta) = \frac{1}{Z_t}\alpha_i(t)a_{ij}\beta_j(t+1)b_j(x_{t+1})$$

– апостериорная вероятность нахождения в состоянии i в момент времени t и в состоянии j в момент времени $t+1$ при условии наблюдаемой последовательности, Z_t – нормализовачная константа, аналогичная по смыслу знаменатели в $y_i(t)$.

После подсчёта этих значений можно с помощью них обновить параметры Θ (M-шаг).

Для нахождения наиболее вероятного пути, генерирующего данную последовательность наблюдений, применяют алгоритм Витерби.

Алгоритм строит наиболее веротяный путь динамическим программированием. Изначально наиболее вероятный путь пустой, вероятность $\pi_t(s)$ – это вероятность наиболее вероятного пути на момент времени t закаанчивающегося в состоянии s. Таблица инициализируется следующим образом:

$$\pi_0(s) = \rho_0(s) b_s(x_0)$$

Далее на каждом следующем шаге $t+1$ таблица пересчитывается из шага t таким образом:

$$\pi_{t+1}(i) = \max_j \pi_t(j) a_i j b_j(x_{t+1})$$

В конце концов в момент времени T будет известна вероятность наилучшего пути, порождающего наблюдения $X = (x_0, \ldots, X_T)$ и его можно будет восстановить по этой таблице, выполняя каждый раз переход по ребру, где достигается максимум.

Пару слов о том, какое пространство состояний можно использовать для HMM в распознавании речи. В первых системах распознавания речи в качестве S использовалось пространство контекстно-независимых фонем, назывемых также монофонами. Быстро стало понятно, что произношение фонемы очень сильно зависит от того, в каком контексте она находится, т.е. от фонем слева и справа от данной. Для решения этой проблемы в качестве пространства фонем стали использовать трифоны $S = \{(q_-, q, q_+)\}$ – помимо центральной фонемы q в тройку состояния также попадают предыдущая фонема q_- и следующая q_+. Но в таком случае быстра возникает проблема с экпоненциальны ростом числа состояний. Например, в русском языке около 50 фонем, тогда трифонов будет $\approx 50^3 = 125000$. У каждого из этих трифонов будут свои параметры для гауссовской смеси распределений и свои переходные вероятности.

Одна лишь матрица переходных вероятностей будет занимать 64GB. Ситуация усугубляется тем, что не все последовательности фонем встречаются в языке, т.е. помимо огромного количества состояний не для всех из них будет достаточно обучающий данных, чтобы восстановить параметры. Для решения этой проблемы используют state tying. Наборы трифонов, близкие по лингвистическим характеристикам объединяют в одни классы, и параметры модели являются общими для класса трифонов. Классы получаются с помощью кластеризации с применением решающих деревьем. Такие классы схожести трифонов иногда называют сенонами. Например, в русских классических системах распознавания речи используют от 4000 до 9000 сенонов, что сильно меньше троек различных фонем.

Для обучения гибридных нейросетевых моделей на основе скрытых марковских цепей (CD-DNN-HMM) нейросеть учат предсказывать по аудиофрейму и его соседям вероятность сенона: $P(y|x)$. Это отличаетс от вероятности наблюдения $P(x|y = s) = b_s(x)$, используемой в классической HMM. Для перехода от одной к другой используют теорему Байеса:

$$P(x|y) = \frac{P(y|x)P(x)}{P(y)}$$

Здесь $P(x|y)$ – интересующая нас велечина, $P(y|x)$ – выход нейросети, $P(y)$ – априорная вероятность сенона, которая вычисляется по обучающему корпусу. $P(x)$ – априорная вероятность аудио. Она не зависит от y поэтому не требуется для декодирования, но для обучения Baum Welch без неё не обойтись. Поскольку её невозможно точно посчитать, нейросети в CD-DNN-HMM обучают без Baum Welch, вместо этого используется слеующий пайплайн:

1. Тренируется CI-GMM-HMM система

2. На основе неё строятся трифоны, их кластеризация, и инициализация параметров для последующего обучения CD-GMM-HMM системы

3. На основе алгоритма Витерби из CD-GMM-HMM системы строятся выравнивания, далее DNN обучается напрямую предсказывать сеноны по аудио фреймам с помощью cross entropy loss и этих выравниваний

4. Для инференса считаются статистики $P(y)$ по корпусу и далее по формуле выше $P(x|y)$ используется в HMM во время декодирования.

Таким образом, пайплайн для обучения DNN в гибридном ASR подходе является довольно сложным. Упростить его призван connectionist temporal classification loss, о котором пойдёт речь дальше.

1.3.3 Connectionist temporal classification

Как уже говорилось выше, CTC убирает требования о многостадийном пайплайне обучения систем по распознаванию речи и заменяет его маргинализацией по всем возможным выравниваниям, встроенной в саму функцию потерь.

Полная информация присутствует в кандидатской диссертации Алекса Грейвса [21], и в статье [13] а здесь эта информация будет изложена в кратком виде.

Предположим, что задано отображение из звуковых сигналов в последовательности произвольной длины T, состоящих из векторов фиксированной длины k. Другими словами, пусть задана $x = (x_1, \ldots, x_T)$ – входная последовательность векторов акустических признаков, $\forall i : x_i \in \mathbf{R}^k$. Параметр T может меняться от фразы к фразе, k же остается постоянным. Кроме того, задан некоторый словарь лексикона $\mathcal{L}$. Мы будем рассматривать в качестве $\mathcal{L}$ словарь букв, но также возможны другие варианты, например, словарь слогов, фонем, пар букв и т.д. Для русского языка $\mathcal{L} = \{\text{'а'}, \text{'б'}, \ldots, \text{'я'}, \text{' '}\}$. Символ пробела находится в словаре для того, чтобы отличать границы разделения слов, но в общем случае он не требуется. В общем случае $\mathcal{L} = \{c_1, \ldots, c_m\}$. Далее строится словарь $\mathcal{L}' = \mathcal{L} \cup \{-\}$, где $-$ – специальный символ, называемый бланком

(blank). Этот символ используется в CTC loss для моделирование отсутствия речи (как тишины, как и прочих звуковых эффектов, не связанных с произносимыми словами – шума, музыки, вздохов и т.д.). Кроме того, этот символ используется, когда надо разделить два одинаковы символа, идущих подряд, как например в слове keep.

Как и в случае с акустическими моделями на основе DNN, нейросеть, обученная с помощью CTC loss моделирует условную вероятность токенов при условии аудиофреймов $P(y|x)$. Входная последовательность x подается на вход нейросети, которая на выходе предсказывает матрицу $\gamma_i^j, i \in \{1, \ldots, T\}, j \in \{1, \ldots, m, m+1\}$. То, какая именно архитектура используется для нейросети, находится за рамками данной работы, однако в разделе, посвященным экспериментам, будет упомянуто, какие именно архитектуры использовались нами. Элементы матрицы γ_i^j интерпретируются как вероятности выдать (emit) элемент $c_j \in \mathcal{L}'$ в момент времени i.

Более формально, рассматривается последовательность независимых случайных величин $\pi_1, \ldots, \pi_T$ (которые, однако, зависят от входной последовательности x, тоже имеющей случайную природу). Эти величины принимают значения в множестве $\mathcal{L}'$. Событие $\{\pi_i == c_j\}$ заключается в том, что на входной последовательности x в момент времени i произносилась буква c_j. Вектор $\pi = (\pi_1, \ldots, \pi_T)$ называется путем или выравниванием. Кроме того, задана случайная строка $y \in \mathcal{L}^*$, называемая разметкой (labelling), соответствующая настоящей транскрипции реплики x. Предполагается, что разметка y получается из пути π следующим образом

$$y = \mathcal{B}(\pi) \tag{1}$$

, где функция $\mathcal{B}$ определена с помощью конечного трансдьюсера 1. Функция $\mathcal{B}$ сначала удаляет повторы, а затем бланки. Например, $\mathcal{B}(aaba a - - ac) = abaac$, $\mathcal{B}(c - - aa - - t) = cat$. Тогда

$$P_{AM}(y|x) = \sum_{\pi} P(y|\pi, x) P(\pi|x) = \sum_{\pi} P(y|\pi) P(\pi|x) \tag{2}$$

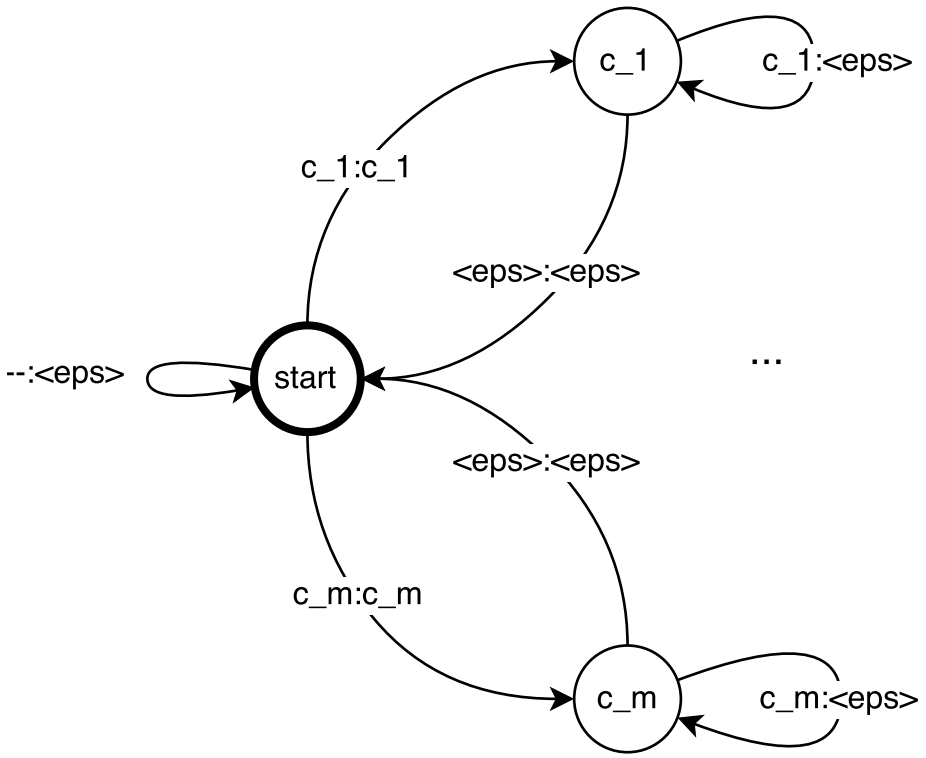

Рис. 1: Finite state transducer для функции $\mathcal{B}$

, где последнее верно, так как y независимо с x при условии π ($y = \mathcal{B}(\pi)$).

Далее, $P(y|\pi) = I\{y = \mathcal{B}(\pi)\}$, поэтому

$$P_{AM}(y|x) = \sum_{\pi \in \mathcal{B}^{-1}(y)} P(\pi|x) = \sum_{\pi \in \mathcal{B}^{-1}(y)} \sum_{i=1}^{T} P(\pi_i|x) = \sum_{\pi \in \mathcal{B}^{-1}(y)} \sum_{i=1}^{T} \gamma_i^{\pi_i} \quad (3)$$

Для упрощения формул мы отождествляем в них случайную величину и ее реализацию.

Нейросеть тренируется так, чтобы максимизировать правдоподобие $\log P(y|x)$, или, что аналогично, минимизировать выражение $-\log P(y|x)$, называемое также CTC loss. Кажется, что множество $\mathcal{B}^{-1}(y)$ очень большое и сложно устроенное, однако это не так и последнюю кратную сумму, и ее градиенты, можно найти с помощью алгоритма forward-backward, который использует динамическое программирование [21], стр. 57.

Опишем forward и backward шаг алгоритма forward-backward, так как они пригодится нам для нахожения лосса, градиентов и реализации дискриминационного обучения в следующей секции.

В графе, изображенном на рис. 2, все пути, начинающиеся в верх-

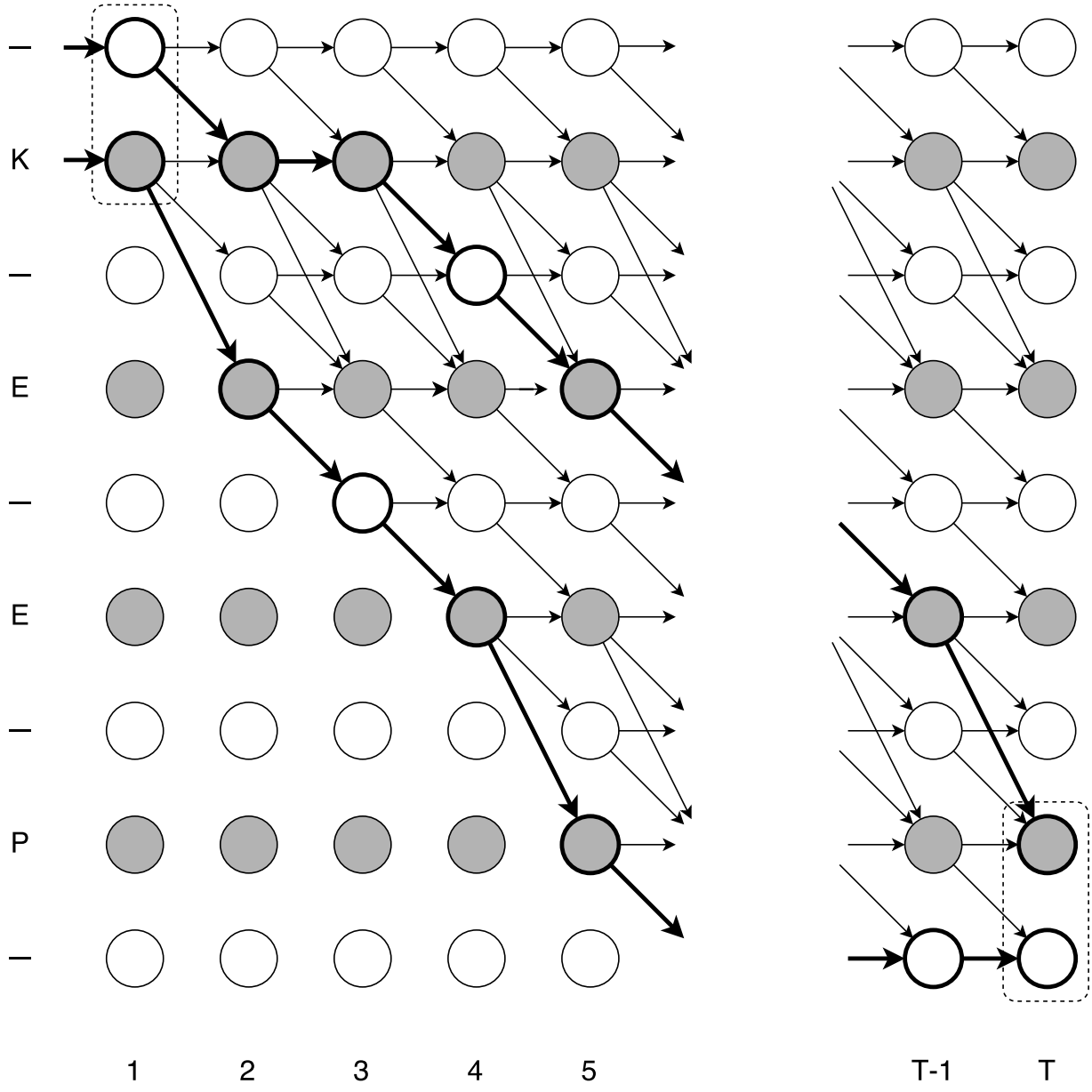

Рис. 2: Возможные пути, дающие разметку $KEEP$. Выделены $KEEP-$ $--\ldots$ и $-KK-E-\cdots-EP$. Заметим, что между двумя строками, соответствующими двум соседним E в KEEP нет ребер, как, например, есть между строками, соответствующими K и первой E, потому что пути, не содержащие хотя бы одного бланка между E, сольются в KEP, а не $KEEP$, после применения $\mathcal{B}$

них левых двух вершинах, и заканчивающиеся в правых нижних, после применения функции $\mathcal{B}$ дадут разметку $KEEP$.

Введем $l' = -y_1-y_2-\cdots-y_n-$. Если $|y| = n$, то $|l'| = 2n+1$. Рассмотрим следующую матрицу $\alpha = (\alpha_i^j), i \in \{1, \ldots, T\}, j \in \{1, \ldots, 2n+1\}$, называемую матрицей forward переменных (или матрицей α переменных), и матрицу $\beta = (\beta^j), i \in \{1, \ldots, T\}, j \in \{1, \ldots, 2n+1\}$, называемую матрицей backward переменных (или матрицей β переменных)

$$\alpha_i^j = P(\mathcal{F}(\pi_1 \ldots \pi_i) = l'_1 \ldots l'_j) \tag{4}$$

$$\beta_i^j = P(\mathcal{F}(\pi_i \ldots \pi_T) = l'_j \ldots l'_{2n+1}) \tag{5}$$

Здесь $\mathcal{F}$ – функция, действующая аналогично $\mathcal{B}$ за тем исключением, что она не удаляет символы blank на последнем шаге. То есть $\mathcal{F}$ по сути удаляет только повторы.

Из рис. 2 видно, что для α и β выполнено следующее рекуррентное соотношение:

$$\alpha_i^j = \gamma_i^j \begin{cases} \alpha_{i-1}^j + \alpha_{i-1}^{j-1}, & \text{if } l'_j = l'_{j-2} \\ \alpha_{i-1}^j + \alpha_{i-1}^{j-1} + \alpha_{i-1}^{j-2}, & \text{else} \end{cases} \tag{6}$$

$$\beta_i^j = \gamma_i^j \begin{cases} \beta_{i+1}^j + \beta_{i+1}^{j+1}, & \text{if } l'_j = l'_{j+2} \\ \beta_{i+1}^j + \beta_{i+1}^{j+1} + \beta_{i+1}^{j+2}, & \text{else} \end{cases} \tag{7}$$

Тогда все α, β переменные можно посчитать за $O(nT)$ проходом по матрице и применением уравнения 6 и 7. Для повышение числовой стабильности все вычисления следует проводить в полукольце логарифмов.

Тогда выполнено следующее соотношение в силу условной независимости элементов π (здесь t – произвольный момент времни):

$$P_{AM}(y|x) = \sum_{i=1}^{2n+1} \frac{\alpha_t^i \beta_t^i}{\gamma_t^{l'_i}} \tag{8}$$

Для нахождения градиентов лосса для обучения методом обратного распространения ошибки продифференцируем это выражение, заметив,

что в числителе терм $\gamma_t^{l'_i}$ присутствует во втором порядке – один раз в α, другой раз в β, и остальные части путей в α и β от него не зависят.

Получается следующая формула:

$$\frac{\partial P_{AM}(y|x)}{\partial \gamma_t^k} = \frac{1}{\left(\gamma_t^k\right)^2} \sum_{i:l'_i=k} \alpha_t^i \beta_t^i \tag{9}$$

Градиент лосса:

$$-\frac{\partial \log P_{AM}(y|x)}{\partial \gamma_t^k} = -\frac{1}{P_{AM}(y|x)\left(\gamma_t^k\right)^2} \sum_{i:l'_i=k} \alpha_t^i \beta_t^i \tag{10}$$

Для численной стабильности рекомендуется пересчитывать знаменатель формулы в 10 с помощью 8

1.4 Языковые модели

Хотя акустические модели, построенные с помощью гибридного или end-to-end подхода можно использовать для распознавания речи без каких-либо дополнительных данных, обычно так не делают. Такие модели ошибаются в написании слов или их окончаний из-за того, что они не имеют никакого статистического представления о языке, с которым работают, кроме тех параллельных данных вида (аудио, транскрипция), на которых они обучались. А собрать параллельный корпус для распознавания речи, сопоставимый по размерам с доступными текстовыми корпусами (сотни миллионов предложений) – непосильная задача.

Для исправления этих недостатков применяют языковые модели. В то время, как акустическая модель моделирует условное распределение $P_{AM}(y|x)$ при условии входных данных $x \in X$ на пространстве всевозможных транскрипций $y \in Y$, языковая модель задает априорное распределение P_{LM} на множестве всевозможных транскрипций и действует в некотором смысле как регуляризатор для акустической модели. Тогда задача распознавания речи заключается в отыскании такой последовательности токенов (слов или букв) $y = (y_1, \ldots, y_n)$, которая максимизирует следующий функционал:

$$Q(y) = \log P_{AM}(y|x) + \alpha \log P_{LM}(y) \tag{11}$$

, где $x = (x_1, \ldots x_T)$ – последовательность из векторов аудио признаков, P_{AM} – распределение, задаваемое акустической моделью, P_{LM} – распределение, задаваемое языковой моделью. Сам процесс отыскания такого y называется декодированием.

1.4.1 n-граммные языковые модели

Языковые модели успешно применяются в системах по распознаванию речи [5] [22]. Большое внимание было уделено n-граммным (n-gram) языковым моделям, которые устроены следующим образом. Выбирается натуральное число n. Обычно $n = 2, 3, 4, 5$. Далее, выбирается большой корпус текста из того же домена. Далее обучается модель, которая пред-

сказывает вероятность $P(w_n|w_{n-1},\ldots,w_1)$ встретить слово w_n при условии его предыдущего контекста слов $w_{n-1},\ldots,w_1$. Из формулы полной вероятности:

$$P(w_n|w_{n-1}\ldots w_1) = \frac{P(w_n\ldots w_1)}{P(w_{n-1}\ldots w_1)} \tag{12}$$

В простейшем случае языковая модель аппроксимирует вероятности в правой части их эмпирическими аналогами, но такая оценка условного распределения игнорирует н-граммы $(w_n,\ldots,w_1)$, которые не встречались в исходном корпусе – для них она эмпирическая вероятность равна 0. Существуют различные техники сглаживания, преодолевающие эту проблему [23].

1.4.2 Декодирование с помощью n-граммных языковых моделей

Декодирование с помощью языковой модели в простейшем случае устроено следующим образом. Рассмотрим для наглядности n=1 и корпус текста, состоящий из 3 различных слов и 3 предложений:

```
K. Cay
K. ache
Cay
```

В этом случае n-граммы в формате arpa будут выглядеть следующим образом:

```
\data\
ngram 1=5

\1-grams:
-0.4259687 </s>
-99 <s>
-0.60206 Cay
-0.60206 K.
-0.9030899 ache
```

```
\end\
```

Число слева от слова в данном формате обозначает логарифм по основанию 10 от вероятности встретить данное слово. <s> и </s> – символы начала и конца предложения. На основе arpa формата генерируется взвешенный конечный трансдьюсер (weighted finite state transducer, WFST) G, который на выход выдает ровно то, что получил на вход, а в качестве веса ребра используются минус натуральный логарифм вероятности перехода из ARPA модели. Пример такого трансдьюсера изображен на рисунке 5

Предположим, что акустическая модель отображает каждый фрейм (небольшой фиксированный промежуток) звука в конечное пространство Z. Например, это может быть пространство фонем, а акустическая модель в этом случае будет являться классификатором фонем. В нашем примере с корпусом из трёх предложений и трёх слов существует две фонемы: k, ey. Лексикон задает отображения из пространства слов W в Z. Лексикон для нашего примера приведен ниже:

```
ache ey k
Cay k ey
K. k ey
```

На основе лексикона строится еще один взвешенный конечный трансдьюсер L, который переводит последовательность классов фреймов из Z и дополнительного символа sil, обозначающего тишину, в последовательность слов из W. Пример такого трансдьюсера представлен на рисунке 3.

Также необходим трансдьюсер токенов T, который преобразует выходы нейросети в пространство $Z \cup \{sil\}$. В простейшем случае классификации каждого фрейма это тождественный трансдьюсер. В случае CTC loss с $Z = \{c_1, \ldots, c_m\}$ трансдьюсер устроен чуть сложнее и изображен на рисунке 4.

Наконец, процесс декодирования заключается в нахождении терминального пути минимальной стоимости с учетом выданных нейросетью вероятностей классов фреймов в детерминированной композиции тран-

сдьюсеров $TLG = T \odot L \odot G$, где T, L, G – трансдьюсеры для токенов, лексикона и языковой модели соответственно.

Данное описание процесса является сильно упрощенным. Подробнее о декодировании можно прочитать в [24]. Идее использования WFST в распознавание речи мы обязаны Mehryar Mohri, который предложил это в своей работе [22]. Подробнее процесс распознавания речи с использованием WFST описан в [25].

Кроме декодирования с помощью WFST, для которого трасдьюсеры должны быть построены в оффлайне и занимают большой объём оперативной памяти при декодировании, при фиксированном T и L (например, как в CTC loss в случае с буквенным лексиконом) и функционально заданном G в виде функции, которая по состоянию декодера и следующему токену выставляет добавочный скор, который этот токен получит можно построить динамический декодер, который строит пути и рёбра в WFST в онлайне. Функционально можно задать G различными способами: например ARPA в виде хеш таблицы, как это сделано в фреймворке KenLM или в виде рекуррентной модели, которая по скрытому состоянию и предыдущему слову выдаёт распределение на следующих словах. Плюсами онлайновых динамических декодером является их пониженные требования к памяти и большая скорость и способность к оптимизации под конкретные T и L.

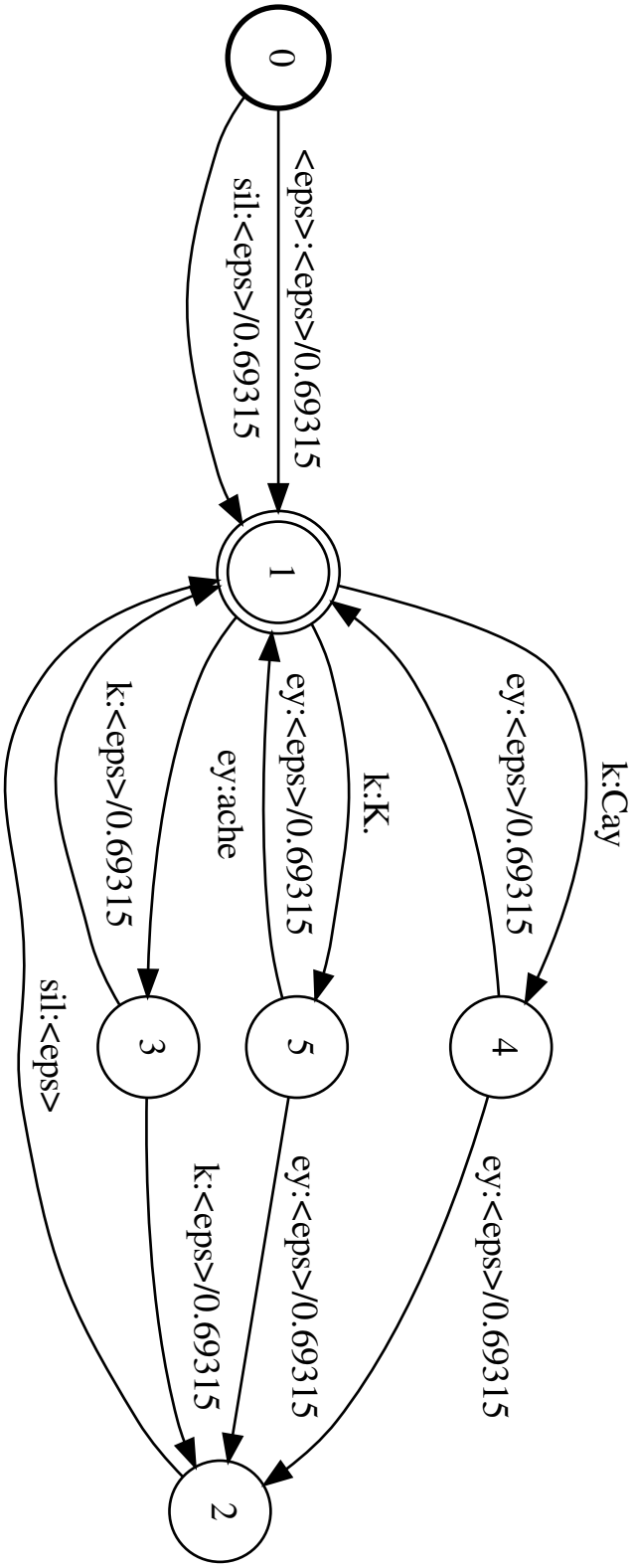

Рис. 3: Пример взвешенного конечного трансдьюсера L для лексикона. Источник: [24]

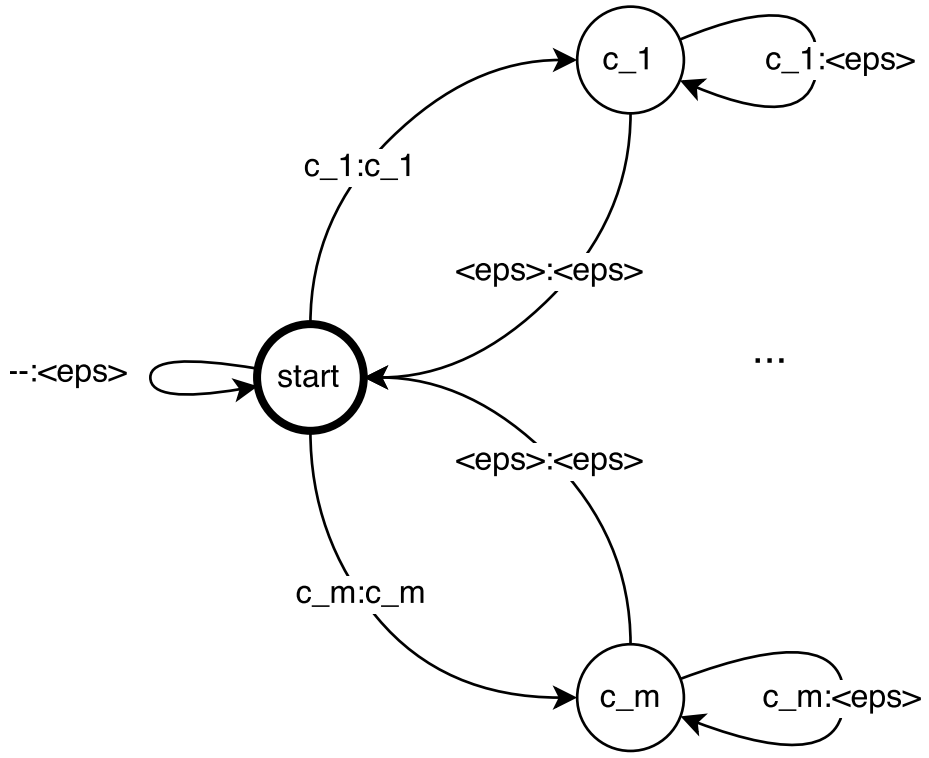

Рис. 4: Пример взвешенного конечного трансдьюсера T для токенов

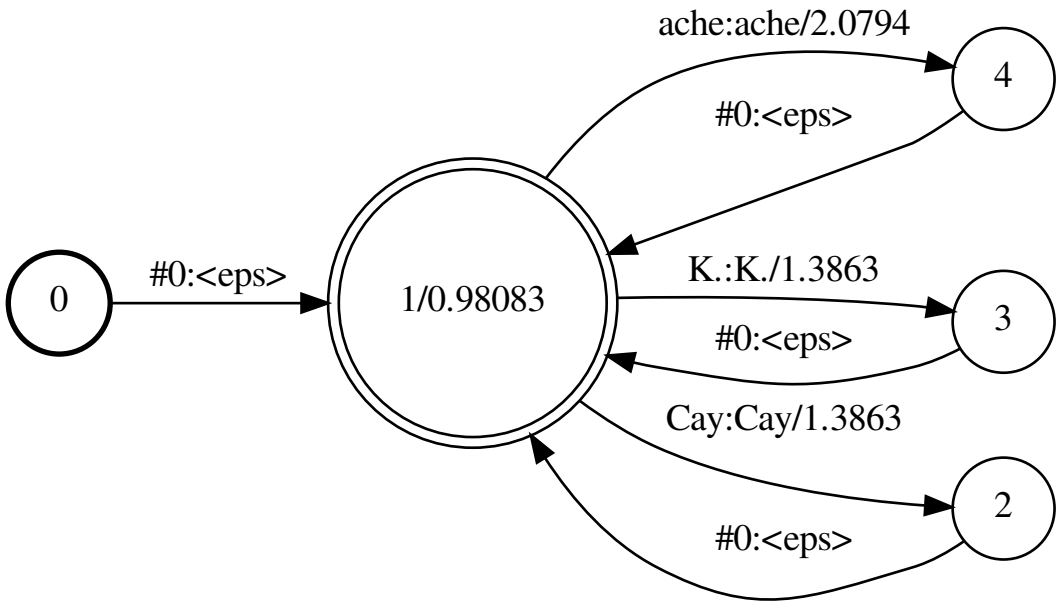

Рис. 5: Пример взвешенного конечного трансдьюсера G для языковой модели. Источник: [24]

1.4.3 Нейросетевые языковые модели

С развитием нейросетей и рекуррентных нейросетевых подходов стали появляться другие методы моделирования языка [26]. В частности, предлагалось заменить контекст из $n-1$ предыдущего слова в n-граммных моделях на некоторый скрытый вектор контекста h и моделировать распределение $P(w|h)$ с помощью рекуррентных нейросетей [27]. Проблема такого подхода заключается в том, что рекуррентную языковую модель трудно встроить в граф декодирования WFST, но можно встроить в онлайновый динамический декодер. Распределение, получаемое рекуррентной языковой моделью (RNNLM) обычно используется для рескоринга (rescoring) списка n лучших гипотез, полученных с помощью классического декодирования, а не для самого декодирования.

С другой стороны, были предприняты попытки построить character-level RNNLM [28], которые моделируют язык побуквенно, и, следовательно, имеют неограниченный словарь, в отличие от word-level language models с фиксированным словарем. Также они способны хорошо моделировать различные формы слов и варианты окончаний слов, в таких морфологически богатых языках, как русский. В пословных языковых моделях на каждую форму слова нужна своя запись в словаре, что многократно увеличивает его размер и делает сложным нахождение нормировочной константы для последнего слоя модели (softmax). Побуквенные языковые модели лишены этого недостатка, в их словаре содержатся буквы, и они выдают вероятность букв, а не слов, соответственно такие приемы, как hierarchical softmax и noise contrastive estimation [29], успешно применяемые в word-level RNNLMs здесь не нужны.

Кроме RNNLM языковых моделей. в последнее время получают всё большее распространение BERT-like языковые модели, которые являются двунаправленными, или классические однонаправленные языковые модели, основанные на трансформере, например, как языковая модель GPT-2. Они выигрывают у классических методов языкового моделирования (n-граммах и RNNLM) по перплексии на многих датасетах и бенчмарках, поэтому их применение должно давать заметный прирост в качестве.

1.5 Second pass rescoring

Для улучшения качества распознавания после декодирования также применяется техника рескоринга с помощью внешней языковой модели. Пусть есть некоторый декодер, который может как явно хранить WFST граф в памяти, как это присходит в случае с Kaldi, так и неявно, строив его динамически (о таком декодере пойдет речь в следующей секции данной работы). В любом случае из данного декодера после окончания распознавания можно получить список альтернативных гипотез, либо в виде N-best list (в случае с динамическим декодером), либо в виде lattice, в случае с WFST декодером. Практика показывает, что часто в этом списке будут альтернативные написания (см. 6). Для квантизации того, насколько разнообразен N-best лист, полученный из декодера, и насколько он подходит для рескоринга, вводят метрику Oracle WER, равную минимум по всему списку гипотез из wer между гипотезой и референскной транскрипцией. На рисунке видно, что $OracleWer$ падает с $\approx 28\%$ WER для $num\_hyps = 1$ (что эквивалентно просто WER) до примерно 12% WER для $num\_hyps = 256$. Для рескоринга можно использовать более крупные модели, которые занимают больше времени, т.к. в продакшене модель нужно будет прогнать один раз на финальных гипотезах, которых к тому же еще не такое большое число, по сравнению с прогоном на каждый момент времени для каждой гипотезы в биме во время декодирования. Например, для рескоринга можно применять различные BERT-like архитектуры, строить ранжирующие модели, который сравнивают пары гипотез в списке Nbest, использовать различные дополнительные фичи и много другое. Устоявшееся мнение в литературе заключается в том, что с помощью рескоринга можно получить дополнительные 10-15% относительного роста качества распознавания.

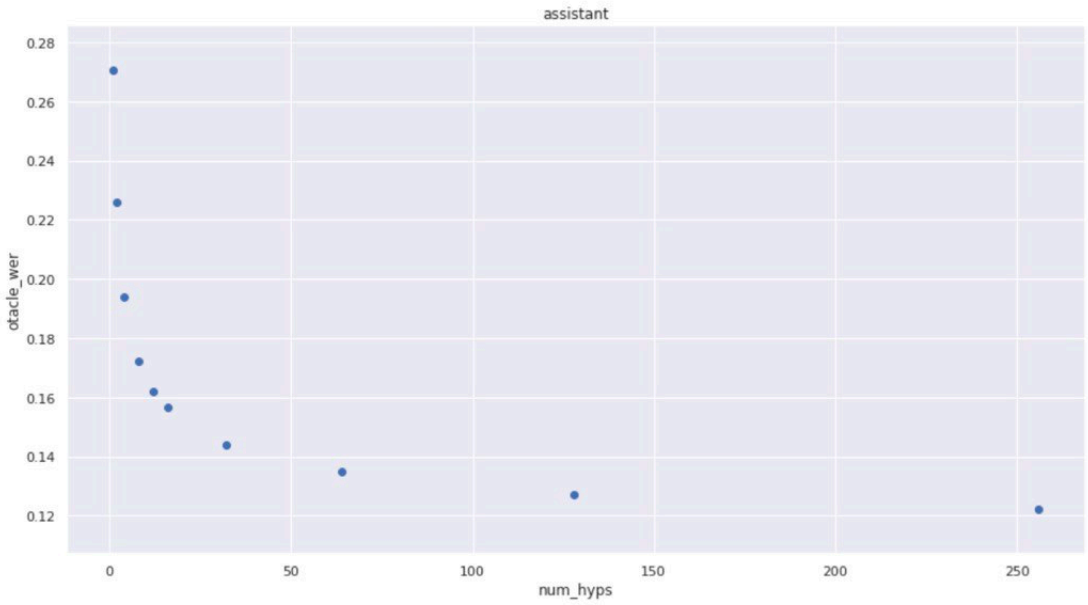

Рис. 6: Orcle wer на одном из валидационных датасетов, в зависимости от количества гипотез в списке N-best гипотез

1.6 LAS и RNN-T

Помимо классического пайплайна, который требует декодирования с внешней языковой моделью, в последние годы стали появляться модели, который способны обучаться и применяться полностью end-to-end. Это означает, что как можно меньше inductive bias было добавлено в систему человеком, и в теории такие алгоритмы должны показывать лучшее качество, чем классичские пайплайны с CD-HMM-DNN или CTC. Одним из примеров такой end-to-end архитектуры являестся Listen, Attend and Spell [11]. Это нейросетевая архитектура, состоящая из 3 блоков: listener, attender и speller. Listener это пирамидальная BiLSTM, котороая отображает последовательность входных аудиофреймов в скрытое состояние. Attender это блок, управляемый декодером, который способен извлекать информацию из скрытых состояний энкодера. Он принимает в себя вектор-запрос Q, с помощью него считает близость к ключам K, выданным энкодером и взвешивает с этими значениями близости значения V, выданные энкодером. Полученный вектор поступает в декодер на следующем шаге. Декодер это UniLSTM сеть, которая генерирует по вектору-кондишену из Attender-а и предыдущей сгенерированной по-

следовательностью транскрипций авторегрессионно следующее слово в транскрипции.

Применяются в продакшене такие модели вместе с beam search decoding, который выдаёт списка топовых гипотез, которые затем рескорятся отдельной языковой моделью, обученной на очень больших корпусах текста. Без последнего шага не удаётся получить качество сравнимое или незначительно лучше классических ASR систем. Кроме того, для получения высокого уровня качества также приходится применять различные трюки, такие как scheduled sampling и minimum wer training, очень тесно связанный с дискриминационными критериями, о которых пойдёт речь ниже.

Ещё одной проблемой LAS является его неспособность работать в онлайне. Как минимум энкодер представляет из себя двунаправленную рекуррентную сеть, так что ему нужна вся последовательность перед тем, как он сможет посичтать скрытое представление. Но даже если бе декодер был однонаправленным, возникает проблема с attention, для его подсчёта всё равно необходима вся последовательноть. Если различные техники получения монотонного аттеншена, один из них Monotonic Chunkwise Attention (MoChA), но такие модели теряют в качесте.

Помомо монотонного аттеншена есть и другой способ превратить LAS в онлайн модель, который был придумал Алексом Грейвсом, который впервые написал о CTC loss, в 2012 году, за 2 года до появляения понятий мезанизма attention и за 3 года до статьи LAS. Речь идёт о RNN Transducer [12]. В ней также есть энкодер и декодер, называемые тут prediction network и transcription network. Единственное существенное отличие от LAS в том, что механизм внимания вынесен из под контроля декодера и используется только в функции потерь. Кроме того, в отличие от soft alignments в аттеншене, когда маргинализация по разным выравниваниям происходит неявно в вычислениях самой модели, в RNN-T она происходит явно во время вычисления функции потерь. RNN-T также применяется вместе с beam search decoder и гипотезы из него таким же образом можно рескорить сторонней языковой моделью, как и в случае с LAS. Но, в отличие от LAS, RNN-T способен работать в онлайне, ес-

ли prediction network является однонаправленной сетью, что даёт ему существенное преимущество в low-latency задачах.

1.7 Метрики качества

При распознавание речи с интегрированной языковой моделью для языков, состоящих из слов, обычно используется метрика WER – word error rate.

$$WER(ref, hyp) = \frac{edit\_distance(ref, hyp)}{len(ref)} \tag{13}$$

, где $edit\_distance$ – расстояние Левенштейна между гипотезой hyp и правильным ответом ref. Здесь считается, что ref и hyp состоят из слов.

Аналогично задается LER – label error rate, только для ref и hyp, состоящих из букв. LER применяется для оценки end-to-end моделей, которые были декодированы напрямую без языковой модели.

К сожалению, LER и WER не дифференцируемы, поэтому их нельзя оптимизировать напрямую с помощью SGD. Вместо этого, как было описано в секции про акустические модели, обычно минимизируют минус логарифм правдоподобия данных из обучающей выборки. Это создаёт mismatch между обучением и применением модели. Побороть этот mismatch как раз призвано дискриминационное обучение и структурное предсказание. Один из вариантов дискриминационного обучения – это минимизация матожидания WER с помщью семплирования Монте-Карло и метода Reinforce, о чём будет рассказано ниже, но существуют также и другие критерии, например MMI, sMBR, MPE. Кроме того, что такие критерии больше похожи на WER, чем классическое правдоподобие, есть ещё один смысл их использовать – они являются sentence-level критериями, по сравнению с CE loss, который является frame-level критерием. Это означает, что дисриминационные критерии при вычислении значения loss и сигнала ошибки учитывают различные конкурирующие гипотезы. Скор плохих гипотез понижается, хороших – повышается. Такой подход к обучению ставится в противопоставление методу максимального правдоподобия, которые рассматривает только одну правильную гипотезу, но не остальные.

Кроме того, для realtime speech recognition используется метрика real

time factor (rtf), равная отношению времени декодирования фразы к длине фразы в секундах. $rtf < 1$ означает, что декодирование можеть происходить в real-time, если алгоритм это позволяет.

2 Декодирование

В этой секции будет описан онлайновый динамический декодер для акустической модели с топологией выходов, эквивалентной той, что применяется в CTC loss. Такой декодер по сложности отличается от beam search декодеров для LAS моделей, в первую очередь за счёт того, что правила по объединению различных выравниваний одной транскрипции, которые в WFST фреймворке задаёт автомат T.fst, а также правила по формированию слов из токенов, которые выдаёт неросеть (в классическом ASR эта часть трансдьюсреа носит название лексикона, L.fst) задаются вручную в коде декодера. В связи с этим такие декодеры дают больше простора для оптимизаций, но являются более сложными в реализации. Например, для хранения лексикона динамические онлайновые декодеры требуют дополнительные структуры данных с быстрой асимптотикой по восстановлению индексов слов по префиксу.

В отличе от этого, декодеры LAS осуществляют поиск по простому древесному графу, изображённому на рисунке 7, с помощью beam search. Такой способ декодирование также называется prefix search decoding. В таких графах не требутеся осуществлять слияние различных путей, и дополнительных скоров из языковых моделей, поэтому они проще в реализации. Декодирование с подобными графами для CTC loss требуют подсчёта и последующего переподсчёта вероятностей префиксов, который выполняется за время $O(NT)$ для последовательности акустических фичей длины T и транскрипции длины N. Эксперименты в наших предыдущих работах показали, что это не даёт существенного прироста в качестве, как можно было бы ожидать, т.к. все активации при обучении с CTC имеют пиковое распределение (т.е. в матрице α вся масса сконцентрирована на диагонали). Тем не менее такое декодирование существенно замедляет скорость работы, т.к. суммарная асимптотика пропорциональна $beam \cdot T^2$.

WFST декодеры устроены чуть сложнее. В отличие от простых beam search декодеров, они нуждаются в шаге рекомбинации путей. Концептуально WFST декодеры устроены аналогично Token Passing Algorithm

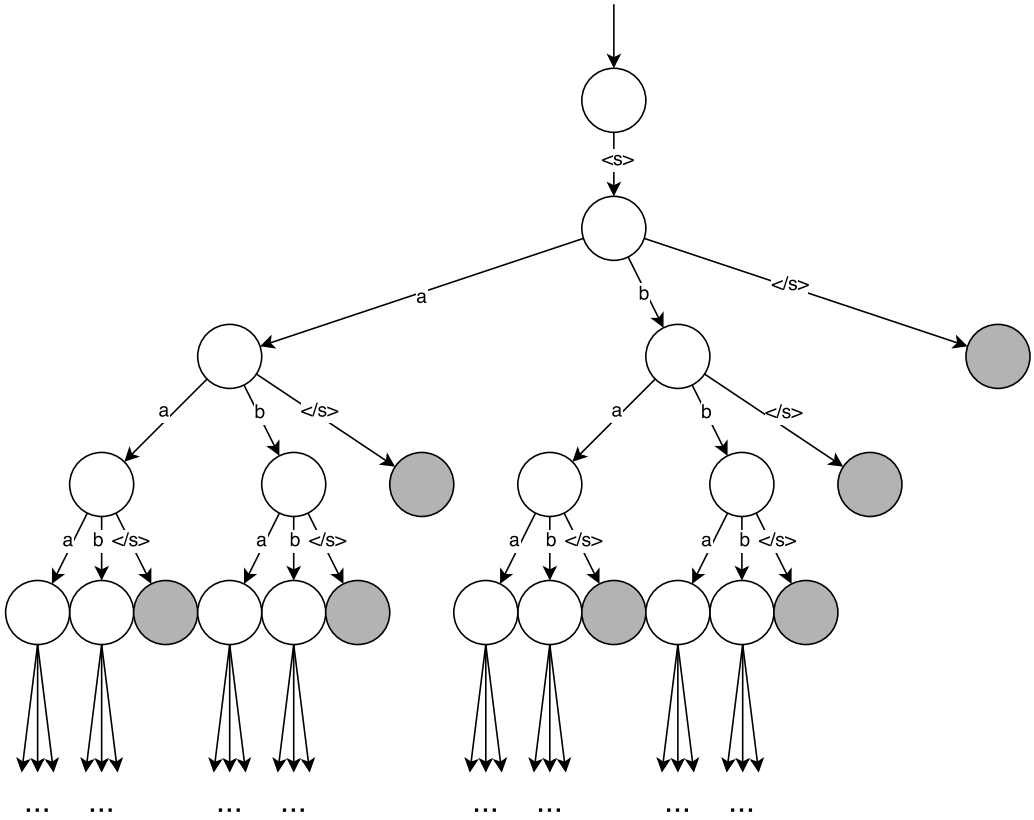

Рис. 7: Граф декодирования для алфавита $\mathcal{L} = \{a, b\}$. Серым помечены терминальные вершины. <s> и </s> – специальные символы, не содержащиеся в алфавите, означающие начало и конец реплики. Корневая вершина расположена сверху

[30], но с различными эвристиками и отсечениями, чтобы ограничить пространство поиска.

2.1 Постановка задачи декодирования

Для постановки задачи декодирования введём понятие полукольца (semiring). Множество R с бинарными операциями $\cdot$ и $+$ называется полукольцом $(R, +, \cdot)$, если выполнено:

- $(R, +)$ – коммутативный моноид с нейтральным элементом 0:
  1. $(a + b) + c = a + (b + c)$
  2. $a + 0 = 0 + a = a$
  3. $a + b = b + a$

- $(R, \cdot)$ – моноид с нейтральным элементом 1:
  1. $(a \cdot b) \cdot c = a \cdot (b \cdot c)$
  2. $1 \cdot a = a \cdot 1 = a$
- Умножение справа и слева дистрибутивно относительно сложения:
  1. $a \cdot (b + c) = (a \cdot b) + (a \cdot c)$
  2. $(a + b) \cdot c = (a \cdot c) + (b \cdot c)$
- Умножение на 0 деструктивно:
  1. $a \cdot 0 = 0 \cdot a = 0$

Примеры полуколец:

1. $(\mathbb{R}, +, \cdot)$ – действительное поле с естественными операциями сложения и умножения является полукольцом, с нейтральными элементами 0 и 1.
2. $(\mathbb{R}_+, +, \cdot)$ – вероятностное полукольцо, неотрицательные целые числа с естественными операциями сложения и умножения и нейтральными элементами 0 и 1. В качестве множества нельзя выбрать $[0, 1]$, потому что такое множество незамкнуто относительно сложения (но замкнуто относительно умножения). Поэтому приходится рассматривать $\mathbb{R}_+$, т.е. вероятности в вероятностном полукольце ненормированы.
3. $(2^{\{a,b\}^*}, \cup, concat)$ – полукольцо множеств строк с операцией объединения множеств и попарной конкатенацией. Нейтральный элемент относительно сложения – пустое множество, относительно умножения – множество, состоящее из пустой строки.
4. $(\mathbb{R}, LogSumExp, +)$ – логаримфмическое вероятностное полукольцо. Нейтральный элемент относительно сложения – $-\infty$, относительно умножения – 0.

5. $(\mathbb{R} \cup \{+\infty\}, \min, +)$ – тропическое полукольцо с минимумом. Нейтальный элемент относительно сложения – $+\infty$, относительно умножения – 0.

6. $(\mathbb{R} \cup \{-\infty\}, \max, +)$ – тропическое полукольцо с максимумом. Нейтральный элемент относительно сложения – $-\infty$, относительно умножения – 0.

Как и прежде, $X = (x_1, x_2, \ldots, x_T)$ – последовательность акустических фичей, который после прохождения акустической модели превращается в матрицу γ_t^i вероятности на шаге t выдать символ i. Как и прежде, последовательность $\pi = \pi_1 \pi_2 \ldots \pi_T$ называется выравниванием, а получаемая из неё коллапсированием некоторых токенов последовательность $y = y_1 y_2 \ldots y_n$ – транскрипцией.

Задача декодера – найти транскрипцию $y \in \mathcal{L}^*$, минимизирующую следующий функционал.

$$Q(y) = \log P_{AM}(y|x) + \alpha \log P_{LM}(y) \tag{14}$$

$$y^* = \arg\max_y Q(y) \tag{15}$$

Декодеры строят наилучшее решение y^* постепенно. Поиск может происходить в разных пространствах: в пространстве транскрипций (как в случае prefix search decoding для LAS и CTC), так и в пространстве путей (WFST, быстрые CTC декодеры, RNN-T).

Поскольку в этой работе мы сравниваем CTC бейзлайн, дальше будет описан метод работы декодеров в пространстве путей. Пусть зафиксировано полукольцо самих путей (аналогичное $(2^{\{a,b\}^*}, \cup, \cdot)$) и их скоров (например, вероятностное или тропическое полукольцо). Декодер в любой момент времени хранит список наилчших гипотез на текущий момент. Но хранить сами пути затратно, учитывая что таких путей может быть экспоненциальное число для каждой гипотезы (некоторые пути могут давать одну и ту же транскрипцию). Поэтому динамические декодеры хранят список транскрипций (классов эквивалентности путей по отношению к операции коллапсирования), и дополнительную метаинформацию,

которая позволяет понять, каким образом их транскрипции можно породить новые пути (для CTC это бланковая и небланковая вероятность). WFST декодеры хранят lattice (ограничение полного графа на список состояний, просмотренный во время декодирования) и список текущих активных состояний. Вместе с гипотезой хранится также и скор, который представляет сумму скоров всех путей (в терминах полукольца), который порождают данную гипотезу.

Пути строяться итеративно, на каждом шаге декодер рассматривает очередной шаг t и вероятности порождения γ_t^i. Текущие гипотезы и пути, им соотвтествующие продлеваются всеми возможными способами. Вероятность каждого из путей домножается (в терминах полукольца, т.е. на пример для тропического полукольца умножение здесь на самом деле обозначает сложение) на γ_t^i, и, возможно, на скор из внешней языковой модели, если таковая имеется (в случае WFST домножается на скор на ребре). В силу дистрибутивности умножения относительно сложения это эквивалентно тому, чтобы скор всей гипотезы домножить на нужное число. После этой операции некоторые пути могут приводить в уже существующие гипотезы, а некоторые – создавать новые. Если путь прходить в уже существующую гипотезу, её скор необходимо рекомбинировать с тем скором, который уже написан в данной гипотезе, для этого необходимо воспользоваться операцией сложения из полукольца. Если путь переходит в новую гипотезу, в качестве скора гипотезы нужно использовать скор пути. Дальше используются различные эвристики для сжатия количества гипотез, например, одни из самых популярных:

1. На каждом шаге оставлять только $beam\_size$ лучших гипотез
2. На каждом шаге оставлять только те гипотезы, которые отличаются от лучшей не больше, чем на $beam$
3. Использовать только $label\_selection\_size$ лучших токенов на каждом шаге
4. Использовать только те токены, которые отличаются от лучшего не больше, чем на $label\_selection\_margin$.

Другие эвристики будут перечислены ниже. Декодирование останавливается, когда мы рассмотрели все таймстемпы. После этого список гипотез (или lattice для WFST) выдаётся наружу как N-best list. Описанный алгоритм имеет асимптотику $O(beam\_size \cdot T \cdot |\mathcal{L}|)$ по времени и памяти, что отличает его в лучшую сторону от квадратичной асимптотики в prefix search decoding для CTC.

2.2 Плюсы и минусы динамического декодера

Декодирование с помощью онлайнового динамического декодера вместо WFST несёт в себе несколько плюсов. Один из самых очевидных – это размер модели. В процессе декодирвоания вся внешняя языковая модель должна присутствовать в памяти, поэтому её размер играет большую роль (чем меньше, тем лучше). В декодере, представленном в данной статье, используется фреймворк KenLM. Он поддерживает несколько вариантов сжатия модели, один из которых – квантизованная Trie модель. По нашим подсчётам, такая модель занимает на один десятичный порядок меньше, чем ARPA, которая в свою очередь занимает на порядок меньше, чем композиция TLG.fst после компиляции и минимизации. Сравнения размеров различных n-грамных моделей представлено в таблице 10. KenLM и динамические декодер позволяют использовать языковые модели, на два порядка превосходящие языковые модели, используемые для WFST.

Кроме выигрша в размере модели, динамические декодеры позволяют использовать нейросетевые языковые модели, такие как RNNLM, ELMO [32], GPT-2 [33] и в какой-то степени BERT [34]. У таких моделей пространство скрытых состояний бесконечно (GPT-2, ELMO, RNNLM). А у BERT вообще нет такого понятия, как скрытое состояние. Такие модели невозможно использовать в WFST фреймворке, где количество состояний конечно. Динамический beam search decoder в свою очередь не накладывает таких ограничений, и можно хранить скрытое состояние языковой модели вместе с гипотезой в списке лучших гипотез, и пересчитывать его от шага к шагу.

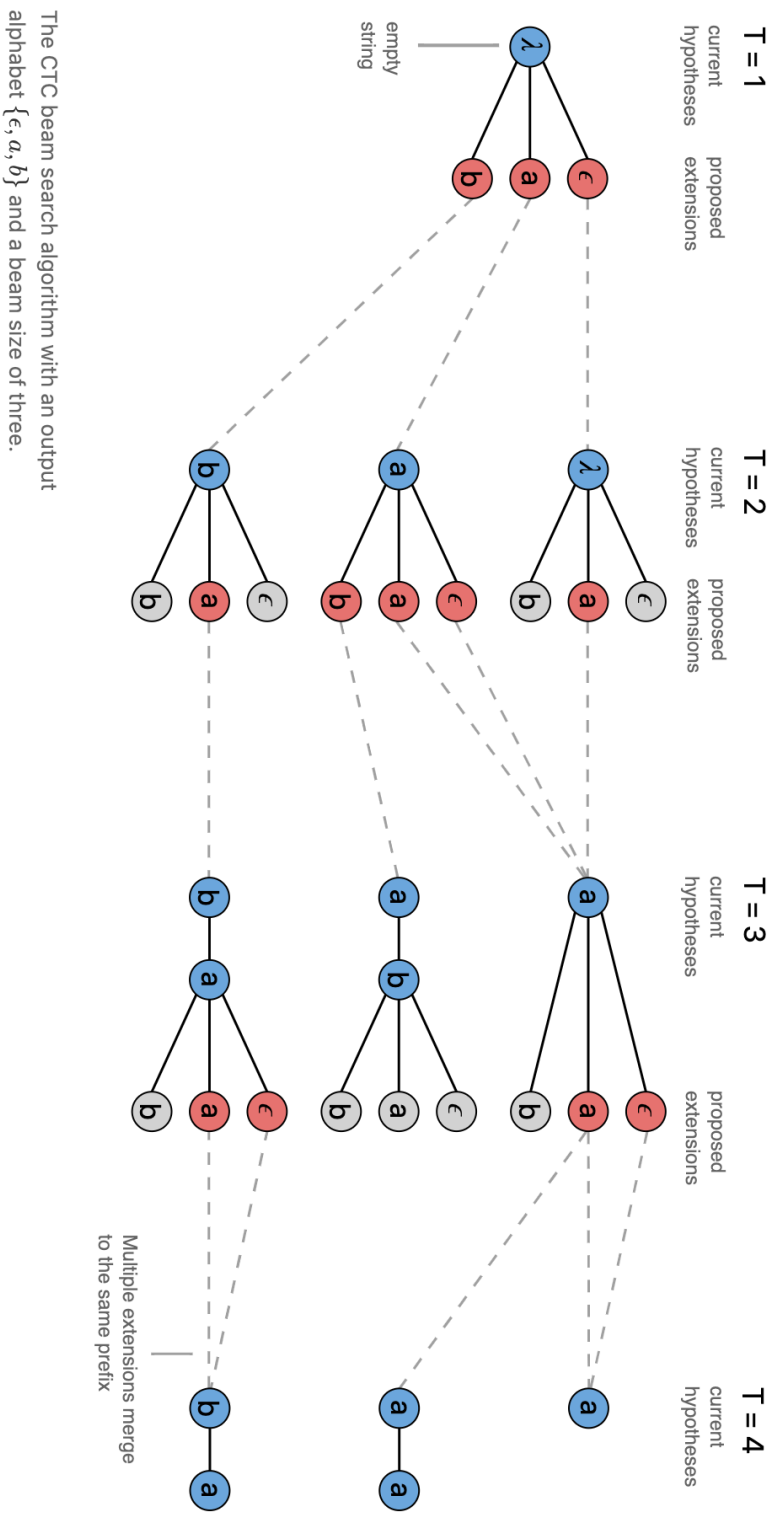

Рис. 8: Иллюстрация к динамическому CTC beam search decoder из [31]. Несколько путей могут сливаться в одну гипотезу.

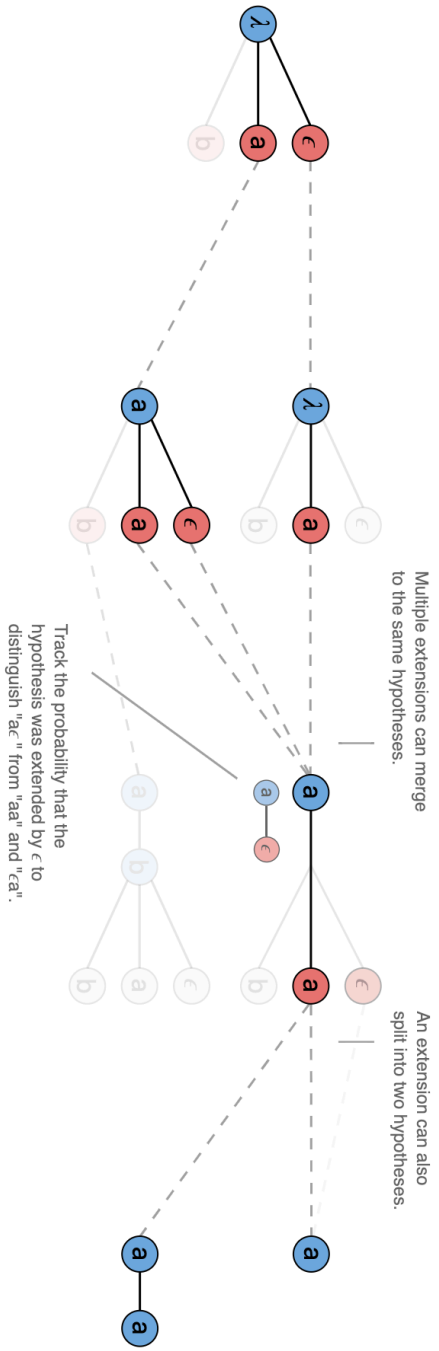

Рис. 9: Иллюстрация к динамическому CTC beam search decoder из [31]. Гипотеза может разбиваться на несколько гипотез. Необходимо отдельно обрабатывать вероятности путей, закачивабщихся на бланк, и не заканчивающихся на бланк

Text source	#tokens	order	prune	ARPA size	TLG size	KenLM trie size
Чаты	366M	4	1	1.6G	6.6G	140M
Чаты	366M	5	1,2	1.8G	14G	160M
Чаты	366M	5	-	17G	-	1.2G
Common Crawl	1.8B	5	-	232G	-	18G

Рис. 10: Сравнение размеров n-граммных модели KenLM, ARPA и WFST для различных текстовых данных и параметров построения. Для построения моделей использовалась утилита lmplz, входящая в поставку KenLM. order – степень n-грамм, prune – параметры, передаваемые в параметр prune в lmplz. ARPA size – размер текстовой ARPA модели, TLG size – размер финального трандьюсера после композиции и минимизации, KenLM trie size – размер KenLM trie модели после квантизации весов.

Помимо этого, динамические декодеры имеют большее простарнство для низкоуровневых оптимизаций, чем WFST. Если в WFST один большой граф загружается в памяти, и дальше необходимо искать пути в этом графе, то в динамическом декодере часть этого графа реализована неявным образом в коде, который можно оптимизировать под конкретные трансдьюсеры T и L. Например, для распознавания с помщью CTC на буквах часть вычислений в этих трансдьюсерах упрощается.

Стоит отметить и плюсы WFST: это их необычная универсальность, такой подход можно применить как к HMM с различными топологиями, так и к CTC без каких-либо изменений в математической теории, а иногда даже и в коде декодера. В случае с динамическими декодерами под каждую пару трансдьюсеров (T, L) необходимо писать кастомный код.

2.3 Описание декодера, используемого в данной работе

Декодер, представленный в этой статье, является динамическим онлайновым декодером с использованием KenLM в качестве внешней языковой модели. Тривиальный лексикон (маппинг из последовательностей букв в слова) хранится в боре, который может быть загружен в виде memory mapping с диска, как и сама языковая модель. В результате нет существенного времени на старт декодера, но может потребоваться время на прогрев, если блоки из файлов языковой модели и бора не лежат в кеше. Декодер написан на C++ и поддерживает расширения различными другими языковыми моделями, в том числе и нейросетевыми, но в данной работе рассматриваются только n-грамные. Гипотезы хранятся в виде дерева, где каждый ребёнок добавляет определённую букву, написанную на ребре, к своему родителю. Для организации структуры TopN для хранения топовых гипотез используется вектор, который амортизировано частично сортируется с помощью std::nth_element. В качестве аллокатора используется jemalloc, который показал существенный прирост в производительности по сравнению со стандартных аллоктором glibc. Чать вычислений (в основном LogSumExp) векторизована с помощью AVX/AVX2 инструкций.

Декодером поддерживаются следующие эвристики:

- LM smearing (min unigram heuristic)
- Beam pruning
- Label selection
- Штраф за слова не из словаря
- Поощрение слов из словаря

2.4 Экспериментальные результаты

Сравнение по скорости и качеству с декодером из Kaldi приведено в 11. Языковая модель для FasterDecoder и Libdecoder была построена на ос-

Decoder	LM weight	Beam	Word reward	Time, s	WER, %
FasterDecoder (Kaldi)	0.5	16.0	-	24.8	17.9
Libdecoder (Ours)	0.65	100	2.5	10.2	14.7
Libdecoder (Ours)	0.65	500	0.3	37.6	13.65
Libdecoder (Ours)	0.65	1000	2.5	210.9	13.0
Libdecoder (Ours)	0.65	2000	2.5	437.8	12.6

Рис. 11: Сравнение WFST декодер из Kaldi и предложенного динамического декодера. Kaldi decoder использует запруненную 5-грамную модель, libdecoder – полную модель, без pruning. Параметр beam в случае libdecoder означает количество лучших гипотез, остающихся после каждого шага, а в случае с FasterDecoder – максимальное отклонение от топовой гипотезы из бима.

нове одного и того же текста, но использовала разные параметры для прунинга. Для libdecoder прунинг не использовался и финальная модель была экспортирована в формат KenLM probing и занимала 5.5G. Для FasterDecoder был построен TLG WFST на основе запруненной 5-грамной модели с параметрами прунинга $1, 2$. Финальный скомпилированный FST занимаал 14.0G. Как видно из графика, libdecoder работает быстрее и с большим качеством, нежели FasterDecoder, что объясняется более полноценной языковой моделью. Кроме того, наблюдается масштабирование качества от параметра $beam\_width$, которое наблюдается вполоть до значений ~ 2000. Таким образом, при одинаковых ограничения на размер используемой памяти и скорость декодера, libdecoder демонстрирует лучшее качество, чем FasterDecoder.

3 Дискриминационное обучение

3.1 Введение в дискриминационное обучение

Как уже упомяналось выше, дискриминационное обучение призвано снизить негативные эффекты расхождения между обучением с помощью Maximum Likelihood критериев, без использования декодера, на одной лишь ground truth гипотезе и инференсом, где используется декодер, несколько гипотез соперничают между собой за первое место в биме и используется недифференцируемая метрика на основе редакторского расстояния (WER, CER). Дискриминационное обучения лучше соответствует принципу максимума апостериорной вероятности, который используется при декодировании. Общепринято мнение в литературе, что такое обучение позволяет добиваться до 5-20% относительного улучшения метрик WER и CER по сравнению с CE или CTC бейзлайном.

Изначально дискриминационное обучение использовалось для HMM-GMM систем. Одним из первых критериев для дискриминиционного обучениея скрытых марковских моделей был MMI критерий (maximum mutual information). Он был успешно применён для задачи phoneme recognition на корпусе TIMIT [35] [36]. Суть MMI обучения состоит в максимизации взаимной информации последовательности аудио признаков и последовательности фонем (состояний в HMM). Если расписать производную от MMI критерий, в ней будет 2 слагаемых. Первое – повышаюшее скор пути (путей) в HMM, соотвтествующих правильной транскрипции, второе – понижающее вероятность всех остальных путей. Эта общая тема с двумя слагаемыми так или иначе прослеживается во всех дискриминативных критериях. Отчасти этим обусловлено их название, в их способности разделять похожие гипотезы между собой. Это же помогает во время инфереса с помощью декодера – гипотезы, которы раньше слабо различались по скору, после дискриминативного обучения разделяются большим гэпом в скоре.

Другой пример дискриминативного лосса – bMMI (boosted MMI), в котором конкурирующие гипотезы взвешиваются с весом $e^{-b\cdot(1-\mathrm{WER}(y,y*))}$. Это позволяет меньше штрафовать гипотезы, которые близки по WER

к истинной гипотезе. MPE [37] (minimum phone error) – ещё один критерий, предложенный Daniel Povey для дискриминационного обучения. По своей формулировке он похож на другой дискриминативный критерий, sMBR [38] (state-level minimum bayes risk), который также используется в ASR. В MPE/sMBR, в отличие от MMI/bMMI, рассматриваются все пути в первом слагаемом, но они взвешиваются с весом PhoneAccuracy$(y, y*)$ для MPE и StateAccuracy$(y, y*)$ для sMBR.

Таким же образом, как декодирование с GMM-HMM и Viterbi на HMM топологии переносится на DNN-HMM и beam search на WFST, дискриминативные критерии также способны задействовать языковые модели похожим образом. В формуле для MMI/bMMI/MPE/sMBR, в месте, где используются пути в HMM, вместо этого можно использовать пути в трансдьюсере. Далее, аналогично тому, как в BaumWelch считаются мат.ожидание занятости (expected occupancy) состояний и ребёр HMM $\gamma_i(t)$ и $\xi_{ij}(t)$, аналогичные параметры можно посчитать для числителя и знаменателя дискриминативных лоссов (с учётом взвешивания путей в WFST). Не основе этих значений считается градиент функции потерь.

На практике state-of-the-art результаты в гибридном ASR достигаются как раз с помощью дискриминативных криетриев. Базовая сеть, натренированная с помощью CE/CTC/LAS/RNN-T дообучается с помощью одного из критериев. Например, для CD-DNN-HMM дообучение даёт 12% прироста для чистого датасета (Switchboard) и 6% для noisy датасета (CallHome) [38]. В своей работе мы наблюдаем улучшение до 12%, большая часть которого сфокусирована на шумных валидационных датасетах. Во время дискриминационного обучения для WFST обычно используют униграмные или биграмные модели, натренированные на акустическом датасете, т.к. они меньше по объёму, и в прошлом сообщалось о негативном влиянии порядка n-грамных моделей на результаты дискриминационного обучения. В своей работе мы также исследуем это влияние. Так как наш подход основан на семплировании и динамическом декодере, это даёт возможность использовать очень большие языковые модели для дискриминационного обучения, в том числе нейросетевые мо-

дели. Помимо гибридных DNN-HMM и CTC подходов, дискриминационное обучение может использоваться также и для LAS моделей, например в статье Minimum WER training [39]. Кроме speech recognition, дискриминативное обучение распространено и в других областях machine learning, например, в neural machine translation, где обученые CE системы дообучают с помощью expected BLEU лосса [40]

3.2 Структурированное обучение и предсказание

Область machine learning, где исследуют предсказания систем, на выходы которых наложены некоторые ограничения, называется структурным обучением и предсказанием. Например, структурным предсказанием является вывод синтаксического дерева на основе предложения. Или задача Part of Speech tagging [41], где выходными токенами является последовательность тегов.

Задача распознавания речи по своей постановке является структурным предсказанием. Ещё со времён применения HMM для распознавании речи в 1980-ые и 1990-ые годы высказывалось предположение о том, что сокращение различий между инференсом и обучением положительно влияет на качество распознавания [42]. К концу 90-ых – началу 2000-ых дискриминативное обучение стало де фактом стандартом для достижения state-of-the-art результатов с HMM. А с приходом глубоких нейронных сетей в распознавание речи и вместе с ними нового state of the art, дискриминационные методы прочно вошли в арсенал учёных, занимающихся распознаванием речи.

3.3 Sampling based methods

Здесь будет приведёно описание метода, который эта статья предполагает использовать для дискриминативного обучения. Выкладки во многом повторяют [43] с теми различиями, которыми нам пришлось совершить для поддержки своего пайплайна и декодера.

Рассмотрим вероятностное полукольцо скоров над гипотезами распознавания речи. Пусть π – некоторый путь (выравнивание) для какой-то

гипотезы. Введём $w(\pi|z)$ – ненормированная вероятность пути π. Пусть z_t^i – логиты, которая выдала акустическая модель на шаге t для лейбла i. Тогда $w(\pi|z) = w_{LM}(\pi) \cdot \prod_{t=1}^{T} \exp(z_t^{\pi_t})$. Здесь мы принципиально не рассматриваем нормирование z_t^i на каждом шаге для получения классовых вероятностей γ_t^i. Вместо этого вероятности путей мы будем нормировать глобально, и этот трюк поможет нам в дальнейшем для упрощения функции потерь и градиента. Введём вероятность пути следующим образом:

$$P(\pi|z) = \frac{w(\pi|z)}{\sum_{\pi' \S} w(\pi'|z)}$$

Вероятность транскрипции (лейбеллинга):

$$P(y|z) = \sum_{\pi:B(\pi)=y} P(\pi|z)$$

Рассмотрим теперь градиент логарифма вероятности пути относительно логитов, которые предсказала акустическая модель:

$$\frac{\partial}{\partial z} \log P(\pi|z) = \frac{\partial}{\partial z} \Big(\log w(\pi|z) - \log \Big(\sum_{\pi'} w(\pi'|z) \Big) \Big) =$$

$$\frac{\partial}{\partial z} \log w(\pi|z) - \frac{\sum_{\pi'} \frac{\partial}{\partial z} w(\pi'|z)}{\sum_{\pi'} w(\pi'|z)} = \frac{\partial}{\partial z} \log w(\pi|z) -$$

$$\frac{\sum_{\pi'} w(\pi'|z) \frac{\partial}{\partial z} \log w(\pi'|z)}{\sum_{\pi'} w(\pi'|z)} =$$

$$\frac{\partial}{\partial z} \log w(\pi|z) - \mathbb{E}_{\pi' \sim w(\pi|z)} \frac{\partial}{\partial z} \log w(\pi'|z) \quad (16)$$

$\frac{\partial}{\partial z} \log w(\pi|z)$ является матрицей такого-же размера, как логиты, содержащей единички в тех местах, через которые проходит путь и нули в других местах.

Minimum bayes risk критерии минимизирует ожидаемую функцию потерь от траскрипции-гипотезы и референсной транскрипции:

$$\mathbb{E}_{y\sim P(y|x)}L(y,y^*) = \sum_y P(y|x)L(y,y^*) = \sum_y \sum_{\pi:B(\pi)=y} P(\pi|x)L(y,y^*) = \sum_\pi P(\pi|x)L(B(\pi),y^*) \quad (17)$$

Вообще говоря, MBR концентрирует массу вероятности: достаточно гибкая модель, обученная до конвергенции с MBR, назначит вероятность 1 последовательности слов с наименьшей функцией потерь. sMBR (state-level minimum bayes risk) определяет лосс как $L(\pi,\pi^*)$ как количество такие таймстемпов, на которых референсный путь π^* отличается от пути π, выданного моделью. Референсный путь обычно получают с помощью нахождения наиболее вероятного выравнивания (hard alignment) для референсной гипотезы y^*. Ключевое свойство sMBR, которое делает его применение особенно оправданным в $WFST$ декодерах заключается в том, что значение функции потерь $L(\pi,\pi^*)$ может быть аддитивно размазано по рёбрах графа-трансдьюсера. В отличие от этого лоссы, основанные на редакторском расстоянии не декомпозируются аддитивно по рёбрах графа, и для них точные вычисления функции потерь и градиентов невозможны.

Мы будем рассматривать как раз функции потерь, основанные на редакторском растоянии, в частности, в качестве $L(y,y^*)$ мы возмём количество замен, вставок и удалений слов, чтобы привести последовательность слов y в последовательность слов y^*.

Опишем процесс сэмплирования путей при условии фиксированных логитов. Для начала логиты прогоняются через декодер, описанный в секции 2, из которого выдаётся список N лучших гипотез, вместе с различными скорами: суммарным, акустическим, языковым, количеством слов из словаря, количеством oov слов и прочими эвристическими значениями. Далее из списка N лучших гипотез из некоторого скора (например, суммарного) сэмпляться гипотезы с температурой. Здесь мы предполагаем, что вся масса вероятности сконцентрирована на N-best списке, поэтому гипотезы, отличные от тех, что присутствуют в N-best листе, можно проигнорировать. То, насколько качественные и разноообразные

гипотезы выдаёт декодер, сильно влияет на качество дискриминационного обучения. Качество гипотез можно измерять с помощью Oracle WER, а разнообразие – с помощью Self-BLEU скора.

После сэмплирования гипотезы из декодера нужно засемплировать путь из этой гипотезы. Для этого можно воспользоваться forward-backward переменными, которые применяли для расчёта CTC и его градиентов. Напомним, что при фиксированном лейбеллинге l и его расширенной бланками версии l', $\beta_t(i)$ – это вероятность находясь в момент времени t в позиции i в l' выдать весь остальной кусок l. Сэмплинг пути будет осуществляться на том же графе, что изображён на рисунке 2. На каждом ребре присутствует априорная вероятность перехода $\gamma_e = \gamma_t^i$, полученная из логитов. Нас же интересует условная вероятность при условии того, что мы дойдём до конца. Для этого заменим вероятности на рёбрах по формуле $\hat{\gamma}_e = \Big(\beta_{t-1}(i)\Big)^{-1} \gamma_e \beta_t(j)$. После этой процедуры все веса, исходящие из вершины суммируются в 1:

$$\sum_e \hat{\gamma}_e = \sum_e \Big(\beta_{t-1}(i)\Big)^{-1} \gamma_e \beta_t(j) = \Big(\beta_{t-1}(i)\Big)^{-1} \sum_e \gamma_e \beta_t(j) = 1$$

Так как $\beta_{t-1}(i) = \sum_e \gamma_e \beta_t(j)$ по формуле для backward переменных. Кроме этого, выполнен следующий инвариант: изначально мы находимся в вершине, из который достижима терминальная вершина (т.к. это гипотеза из декодера). Далее, в любой момент времени если мы будем переходить по рёбрам с $\hat{\gamma}_e > 0$, мы не попападём в вершину, из который недостежима терминальная. Действительно, пусть пусть в какой-то момент времени $\beta_t(j) = 0$, но $\hat{\gamma}_e > 0$. Тогда это противоречит формуле, по которой были рассчитаны $\hat{\gamma}_e$ либо же $\beta_{t-1}(j) = 0$. Но мы предполагаем, что на предыдущем шаге конечная врешина была достижима. Получили противоречие.

В приведённом доказательстве содержиться алгоритм сэмплирования. Нужно начать из стартовой вершины, и пока не приёдем в терминальную рассматривать все рёбра, исходящие из данной вершины, и переходить по ребру e с вероятностью $\hat{\gamma}_e$. Как только попали в терминальную вершину, нужно остановиться. Такой подход к сэмплированию

путей и гипоетз позволяет нам не строить решётки гипотез явно и не хранить WFST, а значит, позволяет добиаваться большей скорости во время обучения.

Опишем теперь процесс подсчёта лосса и его градиентов. Для подсчёта лосса можно использовать метод Монте-Карло:

$$\mathbb{E}L(y, y^*) = \sum_{\pi} P(\pi|z) \cdot L(B(\pi), y^*) \approx \frac{1}{N} \sum_{i=1}^{N} L(B(\pi_i), y^*) = \overline{L}_{batch}$$

где π_i – это сэмпл пути, описанный по методологии выше, $\overline{L}_{batch}$ – среднее значение лосса по бачу из сэмплов.

Для подсчёта градиента воспользуемся следующим трюком:

$$\frac{\partial}{\partial z}\mathbb{E}L(\pi) = \frac{\partial}{\partial z}\sum_{\pi} P(\pi|z) \cdot L(\pi) = \sum_{\pi} L(\pi) \cdot \frac{\partial}{\partial z}P(\pi|z) =$$
$$\sum_{\pi} L(\pi)P(\pi|z)\frac{\partial}{\partial z}\log P(\pi|z) = \mathbb{E}\Big[L(\pi) \cdot \frac{\partial}{\partial z}\log w(\pi|z)\Big] -$$
$$\mathbb{E}L(\pi) \cdot \mathbb{E}\Big[\frac{\partial}{\partial z}\log w(\pi|z)\Big] \quad (18)$$

Таким образом градиент ожидаемого лосса является ковариацией значений лосса на пути и градиента логарифма веса пути. Несмещённая оценка для ковариации выглядит следующим образом:

$$\frac{\partial}{\partial z}\mathbb{E}L(\pi) \approx \frac{1}{N-1}\sum_{i=1}^{N}(L(\pi_i) - \overline{L}_{batch})\frac{\partial}{\partial z}\log w(\pi_i|z)$$

Именно этот лосс и градиент мы используем для дискримианционного обучения. Он имеет интуитивное толкование, что образцы со значением лосса, превышающими средние, уменьшают логиты вдоль соответствующих путей, а образцы, со значением лосса ниже среднего, наоборот, увеличивают логиты вдоль соответсвующего пути.

Помимо приведённого выше варианта формулы есть и другой. Заметим, что

$$\mathbb{E}\Big[\frac{\partial}{\partial z}\log w(\pi|z)\Big] = \mathbb{E}\Big[\frac{\partial}{\partial z}w(\pi|z)\cdot\Big(w(\pi|z)\Big)^{-1}\Big] = C\cdot\sum_{\pi}\frac{\partial}{\partial z}P(\pi|z) = C\cdot\frac{\partial}{\partial z}\sum_{\pi}P(\pi|z) = C\cdot\frac{\partial}{\partial z}1 = 0 \quad (19)$$

Поэтому

$$\mathbb{E}\Big[L(\pi)\cdot\frac{\partial}{\partial z}\log w(\pi|z)\Big] - \mathbb{E}L(\pi)\cdot\mathbb{E}\Big[\frac{\partial}{\partial z}\log w(\pi|z)\Big] = \mathbb{E}\Big[L(\pi)\cdot\frac{\partial}{\partial z}\log w(\pi|z)\Big] \quad (20)$$

Тогда альтернативная несмещённая оценка выглядит следующим образом:

$$\frac{\partial}{\partial z}\mathbb{E}L(\pi) \approx \frac{1}{N}\sum_{i=1}^{N}L(\pi_i)\frac{\partial}{\partial z}\log w(\pi_i|z)$$

Из этих двух оценок первая лучше по причинам, которые мы опишем ниже.

3.3.1 Связь с Reinforcement Learning

Полученные оценки напоминают алгоритм policy gradient и reinforce для обучение с покреплением и недифференцируемым ревордом. Напомним:

$$J(\theta) = \mathbb{E}_{a\sim\pi(a|s)}R(a)$$

$$\nabla J(\theta) = \frac{1}{N}\sum_{i=1}^{N}(R(a_i) - \hat{R}_{baseline})\nabla\log\pi(a_i|s)$$

, где $\hat{R}_{baseline}$ – некоторый бейзлайновый алгоритм, относительно которого мы хотим получить улучшение. Эксперментально было показано, что вычитание бейзлайна уменьшает разброс в градиентах, что положительно сказывается на обучении. Если сравнить формулу для policy gradient с первой формулой, которая у нас получилась для дискриминационного обучения, можно найти много сходств.

Действительно, задачу по дискриминационному обучению можно переформулировать как Reinforcement Learning задачу. В ней акустическая модель является RL агентом, которая взаимодействует с внешней средой (декодером и внешней языковой моделью), на основе этого взаимодействия генерируется последовательность гипотез, каждая их которых получает реворд $R(a) = -\text{WER}(y, y^*)$. Задачей акустики является подстроиться (дообучиться) под внешнюю среду так, чтобы минимизировать искомый WER между гипотезами и истинной транскрипцией.

Одним из первых о соответствии дискриминационных методов и Reinforcement Learning для end-to-end акустических моделей заявил Алекс Грейвса в статье Towards End-to-end Speech Recognition [44]. В это же статье он описал похожий метод, который был назван expected transription loss. Однако, метод Грейвса не использовал внешнюю языковую модель, и семплировал пути из forward-backward матрицы для CTC напрямую. Это лимитировало улучшение по сравнению с бейзлайном.

Помимо распознавания речи, Reinforcement Learning применяется и для дообучения в задачах Neural Machine Translation. Некоторые результаты перечислены в обзорной статье [40] Важными результатами здесь является [45] .

3.4 Детали реализации

Декодер и MBR loss были реализованы как операции для Tensorflow графа, для бесшовного встраивания в существующий Tensorflow пайплайн по обучению акустических моделей. Вычисления над backward переменными были векторизованы с помощью AVX2/AVX512, т.к. их вычисления и сэмплинг является боттлнеком при использовании сэмплированной функции потерь.

После всех оптимизаций скорость обучения оказалась в 1.5 раза медленне, чем класическое MLE обучение с CTC loss-ом. Это является очень хорошим результатом потому что классическое дискриминативное обучение в калди работает в 5-10 раз медленнее, чем обучение с cross entropy критерием.

Скорость обучения не зависит от сложности языковой модели, поэтому для обучения можно использовать очень большую языковую модель, но мы не заметили, что это даёт прирост в качестве. Это направление остаётся перспективым для дальнейших исследований.

Слой	Весов	Параметры
STFT	0	25 ms, 10 ms, 8 kHz, logmel
2D CONV	0.007 M	(11, 21, 1, 32), stride=1
2D CONV	0.11 M	(11, 11, 32, 32), stride=3
GRU	32.0 M	Bidirectional, hs=800, 5 layers
Dense	0.056 M	35 classes

Рис. 12: Описание слоёв и параметров сети

4 Эксперименты

4.1 Архитектура

В данной работе используется структура, аналогичная DeepSpeech2. На вход подаются logmel спектрограммы, посчитанные на 8kHz аудио с окном 25 мс и шагом 10 мс. Далее следуют 2 2D свёртки со следующими параметрами: ширина окна – 21, высота – 11 и ширина окна окна – 11, высото окна – 11. Страйды для свёрток выбраны следующие: 1 и 3, таким образом суммарная длина последовательности уменьшается в 3 раза. Между каждыми слоями используется слой batch normalization. Завершают свёртки 5 слоёв Bidirectional GRU по 800 юнитов каждый. Схематичное изображение архитектуры можно найти на рисунке 13. Параметры и количество весов для слоёв задано в таблице 12. Общее количество параметров – 32 M. Акустическая модель обучается с помосщью momentum SGD с параметрами $momentum = 0.99$ и $learning\_rate = 3 \cdot 10^{-4}$. Используется экспоненциальное затухание learning rate с шагом раз в эпоху и скоростью затухания 0.833. Используется закрытый датасет в ≈ 15000 часов русской речи, сеть обучается с CTC loss в течение 20 эпох, а затем дообучается дискриминативных критерием (если не сообщено обратное). Для обучения используется Tensorflow, фреймворк для распределённого обучения Horovod и кластер из Tesla V100. Для уменьшения негативного эффекта от большого $batch\_size$ при обучении на нескольких нодах используется $warmup$.

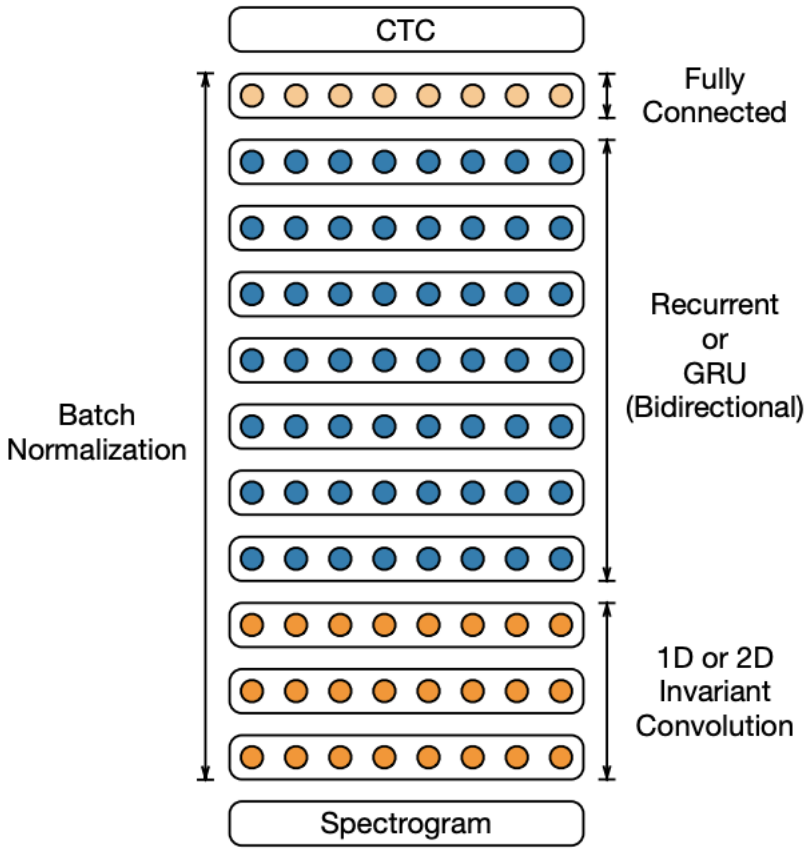

Рис. 13: Схематическое изображение архитектуры, используемой в данной работе. Картинка взята из [15]

4.2 Дискриминационное обучение

4.2.1 MMI и Transcription based minimum WER

Для начала мы попробовали sampling based подход не на основе путей, а на основе транскрипций. Из декодера доставался список N лучших гипотез, они взвешивались с нужными параметрами и от всех этих гипотез во всём баче считался CTC loss. Стандартный $tf.nn.ctc\_loss$ работал медленно и численно нестабильно. Для дискриминативного обучения численная стабильность более важна, т.к. помимо ground truth гипотезы, в сторону которой всё время пробрасывается градиент, в N-best списке присутствуют и более плохие гипотезы, и важно, чтобя для них loss был не NaN и не $+\infty$ и были доступны нормальные градиенты. Поэтому нами была реализована собственная Tensorflow операция (наравне с декодером и path sampling MBR методом), которая во-первых была численно стабильная (использовались вычисления в полукольце LogSumExp), во-вторых – более быстрая. По нашим экспериментам релизованная операция была в 5-8 раз быстрее аналогичной из Tensorflow и выдавала аномальное значение лосса только на транскрипциях, которые не умещались в логиты.

Но сами эксперименты по качеству с дискриминативными лоссами на уровне транскрипций (когда отсутствует шаг сэмплинга путей с помщью патрицы backward переменных) не дали прироста. Наоборот, они оказались негативно влияющими на финальное качество. На рисунке 14 представлены графики с WER на валидационном датасете до и после включения дискриминационной тренировки. WER считался по топ 8 гипотезам из декодера и из него вычитася бейзлайн в виде среднего погипотезам. Видно, что после включения MWER тренировки качетсво распознавания падает. На рисунке 15 изображён flat start training с MMI критерием. Его применение также не дало прироста качества. Мы предполагаем, что потери в качестве и эффект переобучения, отчётливо наблюдаемый на графике MMI, связанны с тем, что transcription based критерии излишне сглаженны ctc loss-ом, который испольется внутри этих критериев для подсчёта градиентов для одной гипотезы. Это рабо-

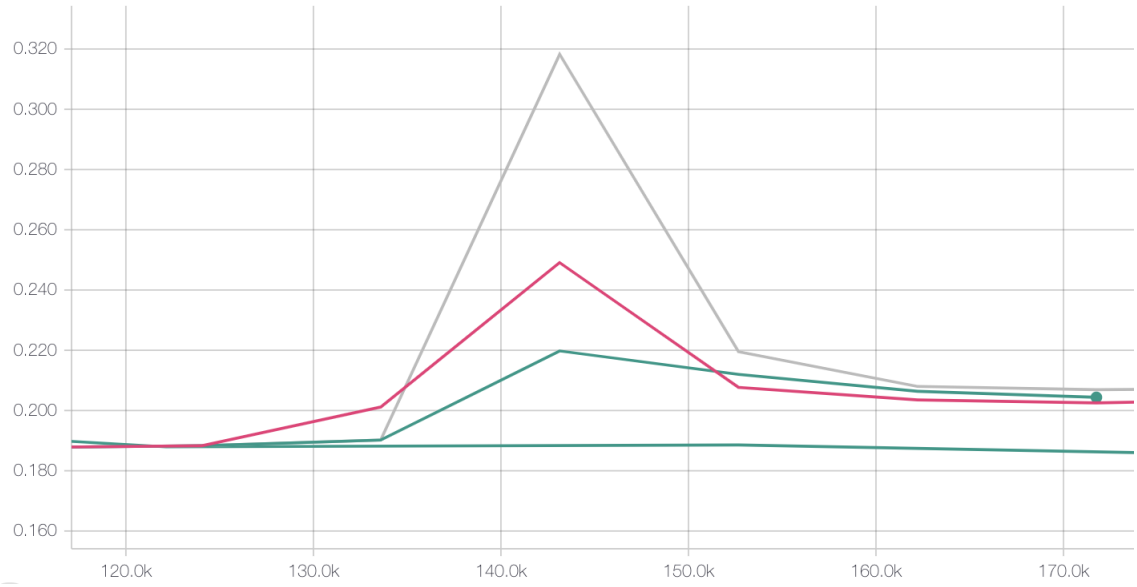

Рис. 14: Изменение WER на валидационном датасете после включения transcription level дискриминационных критериев. Шаг делается раз в эпоху, нижняя прямая – бейзлайн. Верхнии три прямые – MWER с различными параметрами

тает в противопоставление с тем, как используется MWER для LAS и reinforce для задач обучения с подкреплением. В первом случае градиенты сфокусированы ровно на тех местах, где наблюдается несоотвествие выданного лейбла и лейбла из гипотезы. Во втором случае учиться стохастическая политика, которая обладает большим шумом в градиентах.

Перейдём теперь к экспериментам с path-based sampled MBR, в котором эти две проблемы решены.

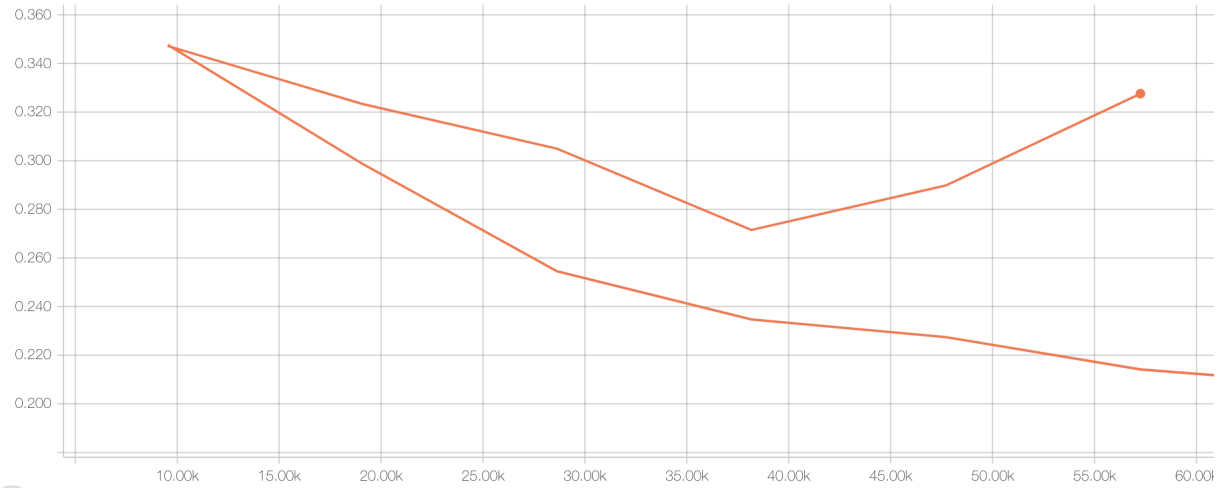

Рис. 15: Изменение WER на валидационном датасете после включения transcription level дискриминационных критериев. Шаг делается раз в эпоху, нижняя прямая – бейзлайн. Верхнии кривая – MMI

4.2.2 Path based minimum WER (sampled MBR)

Sampled MBR, также реализованный нами в виде отдельной операции для Tensorflow, был оптимизирован и подстроен так, чтобы скорость обучения совпадала по порядку с той, что была при использовании CTC loss 16. После компенсации скорости под постоянный прогон dev сета и подсчёта WER мы получили, что скорость sampled mbr методы примерно в 1.5 раза меньше CTC loss.

Для первоначального подбора параметров для sampled MBR использовалась train выборка и метрика WER на ней. Было проведёно 3 сравнительных эксперимента:

1. Температура 4.0 и ограничение на количество гипотез vs температура 2.0
2. $learning\_rate = 10^{-4}$ vs $learning\_rate = 2 \cdot 10^{-5}$
3. $beta = 0.01$ vs $beta = 0$

Параметр β отвечает за интерполяцию sampled MBR лосса и классического CTC лосса. Используется следующая формула:

$$\mathcal{L}(\theta) = \mathcal{L}_{MBR}(\theta) + \beta \cdot \mathcal{L}_{CTC}(\theta)$$

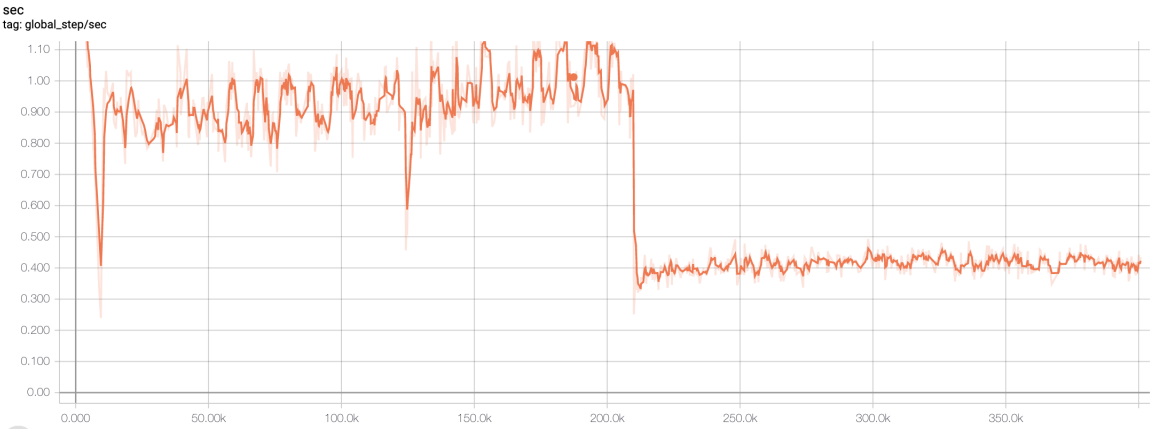

Рис. 16: Скорость обучения с CTC loss и sampled MBR. Помимо смены функции потерь посередине графика дополнительно включилась функция подсчёта wer на dev датасете каждые 100 шагов, что также повлияло на скорость.

Результаты начальных экспериментов можно наблюдать на 17. Из этих данных можно сделать несколько выводов: температура $= 4.0$ не самый удачный выбор, акустическая модель в этом случае расходится даже на трейне. Аналогично можно сказать и про $learning\_rate = 10^{-4}$, хотя и расхождение наблюдается в сильно меньшей степени. Эксперименты по интерполяции с небольшим коэффициентом $\beta = 0.01$ и отсутствие интеполяции идут примерно одинаково, но интерполяция чуть-чуть выигрывает. Можно сделать вывод, что использование интерполяции немного стабилизирует тренировку, делает её более предсказуемой. К такому же выводу прихоядт авторы других статей по дискриминационным критериям. Например, интерполяция с CE критерием носит название F-smoothing и положительно влияет на обучение CD-DNN-HMM.

После первоначальных экспериментов с WER на обучающей выборке нами был составлен dev датасет из 16K примеров из разных валидационных выборок. В отличие от валидационного датасета, метрики на котором считаются раз в эпоху, метрики на dev датасете рассчитываются каждые 100 шагов. Разумеется, ни валидацонные, ни dev датасеты не пересекаются с трейном по фразам. Далее перейдём к сравнению на dev и valid выборках.

Результаты всех экспериментов на дев сете для sampled MBR пред-

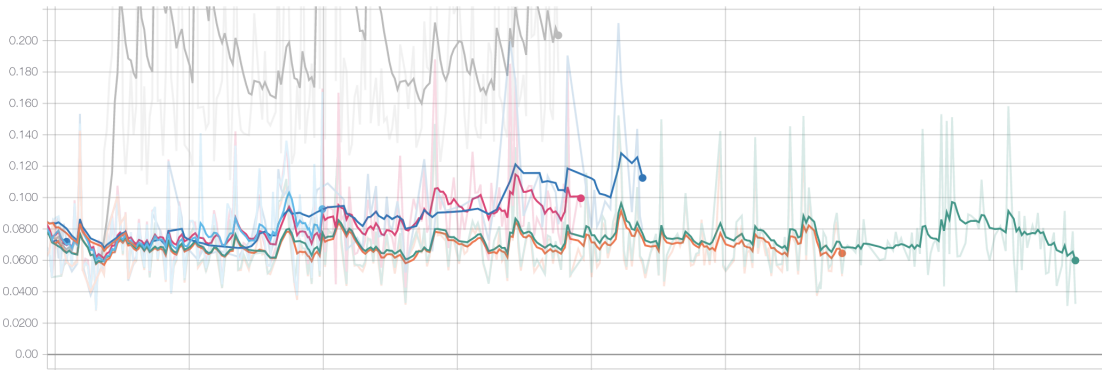

Рис. 17: Изменение WER на тренировосчном датасете для первоначального подбора параметров для sampled MBR. Серая кривая имеет темпаратуру 4. Расходящиеся синяя и розовая линия – learning rate 1e-4. Нижние две линии имеют lr=2e-5, и темпаратуру 2, их отличие в том, что та линия, что идёт ниже, дополнительно используют интерполяцию с CTC с коэффициентом $beta = 0.01$

ставлены на рисунке 19. Все графики старуют из точки WER=18.1. Чтобы сравнить с обычным ctc loss, который до этого не обучался с dev метриками, мы стартовали его обучение с новыми метриками на дев сете, и наблюдали колебания WER в промежутке от 17.5% до 18.5%. Это означает, что sampled based MBR даёт прирост от 8% до 12% качества относительно CTC бейзлайна. Далее подробно рассмотрим каждый их экспериментов.

Две запуска не представлены на этом графике, т.к. они разошлись. Это эксперимент с batch_size=30x8 вместо 96x8. Уменьшение бач сайза в 3 раза эквивалентно увеличению $learning\_rate$ в 3 раза, что подтвердило нашу докадку, озвученную выше, что sampled MBR более чувствителен в выбору начальных параметров, нежели CTC, в том числе к скорости обучения. Второй эксперимент, который разошёлся, использовал с lm weight = 1.0 вместо 0.5.

На рисунке 20 представлен график WER на дев сете в зависимости от learning rate. Не смотря на то, что learning rate = 10^{-5} вначале отставал от двух других конкурентов ($2 \cdot 10^{-5}$ и $4 \cdot 10^{-5}$), потом он их нагнал и в итоге сошёлся к лучшему минимуму. Сеть, обучаемая с $learning\_rate = 4 \cdot 10^{-5}$ начала переобучаться после первой эпохи. Та-

	dev	dictation	queries noisy	calls noisy
Baseline	18.1	5.6	28.4	18.2
Default MBR	16.17	5.2	24.4	17.8
+ 5gram	16.21	5.1	24.4	17.9
+ beta=0.02	16.16	4.9	24.2	17.6

Рис. 18: Сравнение различных параметров sampled MBR на валидационных выборках

ким образом, оптимальным learning rate для sampled MBR лосса является 10^{-5}.

На рисунке 21 приведёно сравнение различных параметров beta. Сравнивалось три значения параметра: $\beta = 0.005$, $\beta = 0.01$ и $\beta = 0.02$. Все 3 значения шли в течение первой эпохи примерно одинаково, но после прохождения первой эпохи $\beta = 0.02$ начало расходиться, $\beta = 0.005$ осталось примерно на том же уровне, а $\beta = 0.01$ продолжило снижаться. Видно, что значание $beta = 0.01$ оптимально.

На рисунке 22 представлено сравнение различных языковых моделей – пятиграмной и биграмной. Разницы в скорости обучения не наблюдалось. Существенных различиый а качестве также не обнаружено. Необходимы дальнейшие исселодвания в этом направлении, возможно, лучший подбор параметров декодирования для пятиграмной модели.

Финальные результаты на вадиацонных датасетах представлны в таблице 18. В отличие от дев сета, на валидацонных победила модель с $\beta = 0.02$. Итоговый прирост в WER в зависимости от датасета измеряется от 5% до 15%. На датасетах с более шумными транскрипциями прирост более заметен, т.к. в sampled MBR лоссах референсные транскрипции не присутствуют напрямую, а лишь взвешивают гипотезы, возвращаемые декодером, с помощью WER. Если референсная транскрипция шумная, но всё равно похожа на правильную, монотонность в WER от хороших гипотез к плохим всё равно сохраняется.

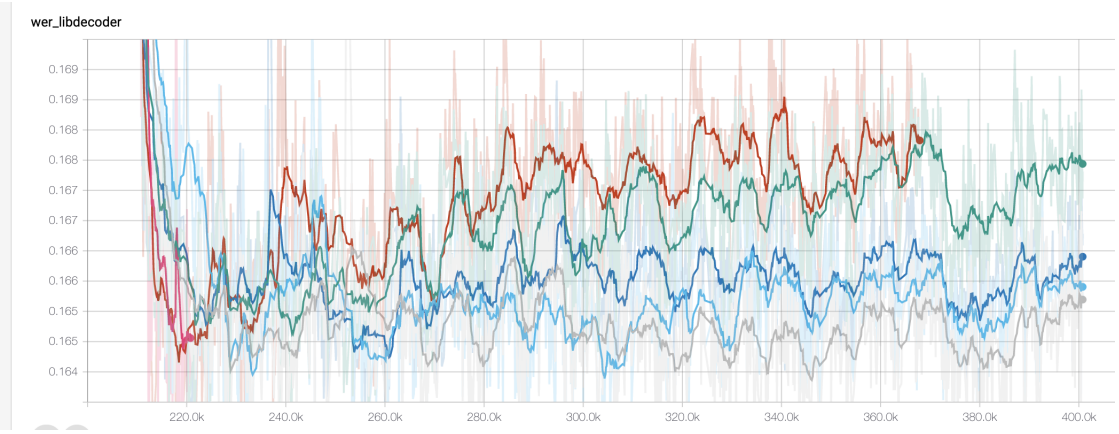

Рис. 19: Изменение WER на дев датасете для подробного подбора параметров для sampled MBR.

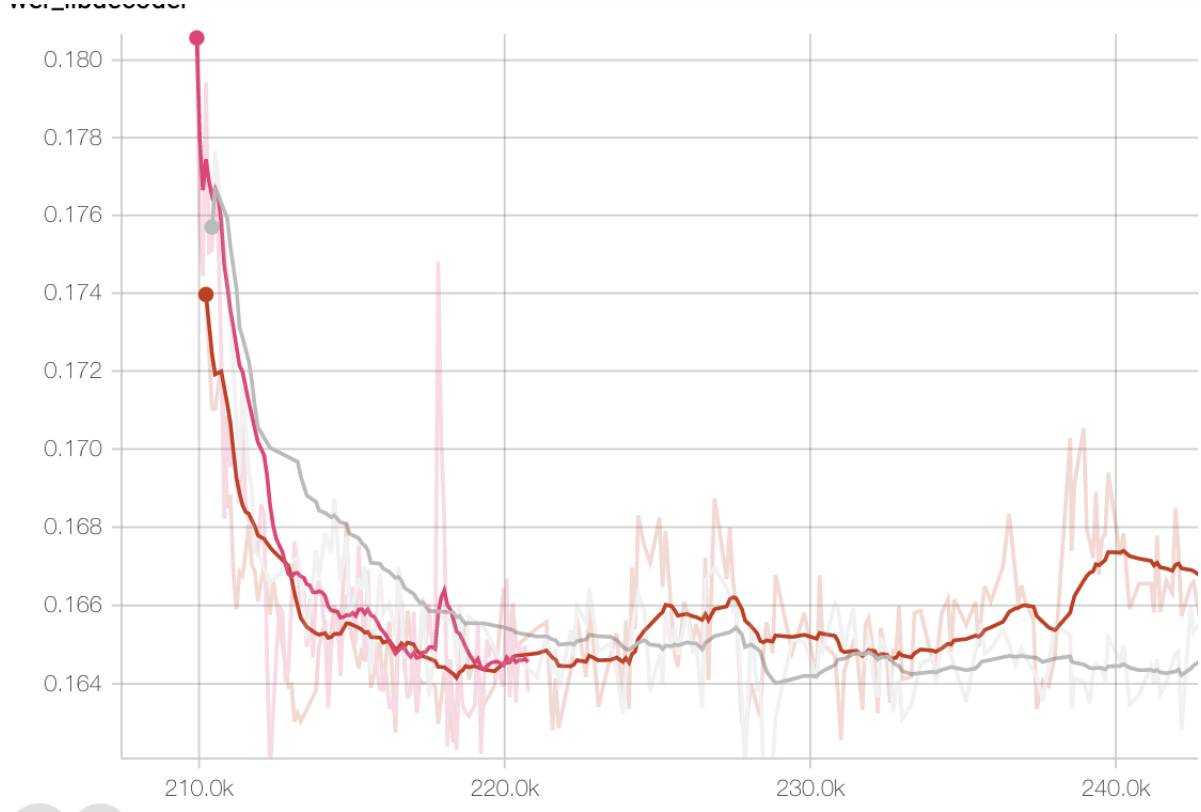

Рис. 20: Sampled MBR с различными значениями learning rate. Серый – 1e-5, розовый 2e-5, коричневый – 4e-5

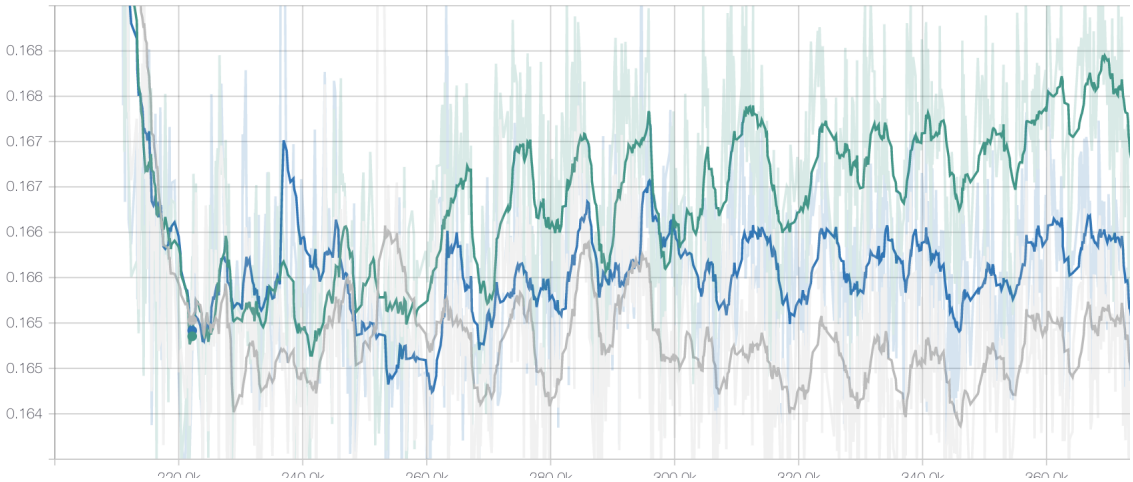

Рис. 21: Sampled MBR с различными значениями beta. Серый – beta=0.01, синий beta=0.005, зелёный – beta=0.02

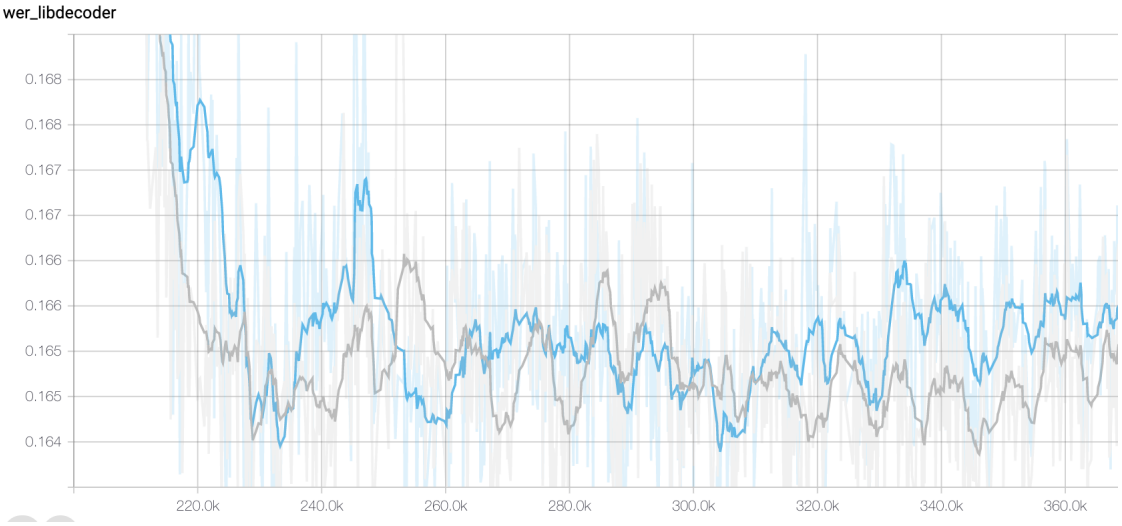

Рис. 22: Sampled MBR с различными значениями языковыми моделями. Серый – биграмная слабая языковая модель на транскрипциях, синяя – сильная 5 грамная модель на разных источниках данных. Однозначный вывод сделать нельзя.

5 Заключение

5.1 Итоги работы

Нами был представлен новый подход к дискриминационному обучению для end-to-end акустических моделей, была проведена параллель к обучению с подкреплением, теоретически обоснованы формулы для градиентов и лосса. Был реализован свой онлайновый динамический декодер на C++ с использованием KenLM и поддержкой нейросетевых языковых моделей, который не требует присутствия всего графа декодирования в памяти и работает в 2-3 раза быстрее WFST декодера, встроенного в Kaldi, а памяти занимает на 2 порядка меньше. Это позволяет использовать большие n-граммные модели с меньшими индуцированными смещениями и натренированные на больших корпусах текста.

Нами была реализована path-based sampled MBR функция потерь, которая напрямую минимизирует математическое ожидание WER – метрики, с помощью которой сравниваются системы распознавания речи. И функция потерь, и декодер были встроены в существующий пайплайн для обучения акустических моделей в виде кастомных опреаций для Tensorflow. Для данной функции потерь были подобраны оптимальные параметры и получено улучшение по сравнению с CTC бейзлайном от 5% до 15% относительных пунктов в искомой метрике в зависимости от датасета.

5.2 Дальнейшие исследования

Дальнейшие продолжения этой работы могут включать – исследования больших языковых моделей для дискриминационного обучения, исследование нейросетевых языковых моделей. Реализация предложенных методов для других типов систем распознавания речи, в первую очередь LAS и RNN-T. Исследование различных технологий сглаживания, помимо интерполяции с CTC loss, например, добавление жёстких выравниваний в градиент с положительным коэффициентом. Дальнейшее исследование MMI и MWER лоссов.

6 Список литературы

1. B.-H. Juang and L. R. Rabiner, "Automatic speech recognition–a brief history of the technology development," *Georgia Institute of Technology. Atlanta Rutgers University and the University of California. Santa Barbara*, vol. 1, p. 67, 2005. [Online]. Available: http://web.ece.ucsb.edu/Faculty/Rabiner/ece259/Reprints/354_LALI-ASRHistory-final-10-8.pdf

2. L. Rabiner and B. Juang, "An introduction to hidden markov models," *ieee assp magazine*, vol. 3, no. 1, pp. 4–16, 1986. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.957.202&rep=rep1&type=pdf

3. X. D. Huang, Y. Ariki, and M. A. Jack, *Hidden Markov models for speech recognition*. Edinburgh university press Edinburgh, 1990, vol. 2004. [Online]. Available: http://www.anthology.aclweb.org/J/J93/J93-1016.pdf

4. J. Benesty, M. M. Sondhi, and Y. Huang, *Springer handbook of speech processing*. Springer, 2007.

5. D. Yu and L. Deng, *Automatic speech recognition: A deep learning approach*. Springer, 2014. [Online]. Available: https://books.google.ru/books?id=rUBTBQAAQBAJ

6. G. Hinton, L. Deng, D. Yu, G. E. Dahl, A.-r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath *et al.*, "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 82–97, 2012. [Online]. Available: https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/HintonDengYuEtAl-SPM2012.pdf

7. A.-r. Mohamed, G. E. Dahl, and G. Hinton, "Acoustic modeling using deep belief networks," *IEEE Transactions on Audio, Speech, and*

Language Processing, vol. 20, no. 1, pp. 14–22, 2012. [Online]. Available: http://www.cs.toronto.edu/~asamir/papers/speechDBN_jrnl.pdf

8. H. Sak, A. Senior, and F. Beaufays, "Long short-term memory based recurrent neural network architectures for large vocabulary speech recognition," *arXiv preprint arXiv:1402.1128*, 2014. [Online]. Available: https://arxiv.org/pdf/1402.1128.pdf

9. I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Advances in neural information processing systems*, 2014, pp. 3104–3112. [Online]. Available: http://papers.nips.cc/paper/5346-information-based-learning-by-agents-in-unbounded-state-spaces.pdf

10. D. Bahdanau, J. Chorowski, D. Serdyuk, P. Brakel, and Y. Bengio, "End-to-end attention-based large vocabulary speech recognition," in *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on*. IEEE, 2016, pp. 4945–4949. [Online]. Available: https://arxiv.org/pdf/1508.04395.pdf

11. W. Chan, N. Jaitly, Q. Le, and O. Vinyals, "Listen, attend and spell: A neural network for large vocabulary conversational speech recognition," in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2016, pp. 4960–4964.

12. A. Graves, "Sequence transduction with recurrent neural networks," *arXiv preprint arXiv:1211.3711*, 2012.

13. A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber, "Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks," in *Proceedings of the 23rd international conference on Machine learning*. ACM, 2006, pp. 369–376. [Online]. Available: http://machinelearning.wustl.edu/mlpapers/paper_files/icml2006_GravesFGS06.pdf

14. A. Hannun, C. Case, J. Casper, B. Catanzaro, G. Diamos, E. Elsen, R. Prenger, S. Satheesh, S. Sengupta, A. Coates *et al.*, "Deep speech: Scaling up end-to-end speech recognition," *arXiv preprint arXiv:1412.5567*, 2014. [Online]. Available: https://arxiv.org/pdf/1412.5567.pdf

15. D. Amodei, S. Ananthanarayanan, R. Anubhai, J. Bai, E. Battenberg, C. Case, J. Casper, B. Catanzaro, Q. Cheng, G. Chen *et al.*, "Deep speech 2: End-to-end speech recognition in english and mandarin," in *International Conference on Machine Learning*, 2016, pp. 173–182. [Online]. Available: https://arxiv.org/pdf/1512.02595.pdf

16. E. Battenberg, J. Chen, R. Child, A. Coates, Y. G. Y. Li, H. Liu, S. Satheesh, A. Sriram, and Z. Zhu, "Exploring neural transducers for end-to-end speech recognition," in *2017 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*. IEEE, 2017, pp. 206–213.

17. S. Davis and P. Mermelstein, "Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences," *IEEE transactions on acoustics, speech, and signal processing*, vol. 28, no. 4, pp. 357–366, 1980.

18. X. Huang, A. Acero, H.-W. Hon, and R. Reddy, *Spoken language processing: A guide to theory, algorithm, and system development*. Prentice hall PTR Upper Saddle River, 2001, vol. 1.

19. D. Yu and L. Deng, *AUTOMATIC SPEECH RECOGNITION*. Springer, 2016.

20. L. Deng, "Computational models for speech production," in *Computational models of speech pattern processing*. Springer, 1999, pp. 199–213.

21. A. Graves, "Supervised sequence labelling with recurrent neural networks," Ph.D. dissertation, Technische Universitat Munchen, Fakultat

fur Informatik, Lehrstuhl VI: Echtzeitsysteme und Robotik. [Online]. Available: http://www.cs.toronto.edu/~graves/phd.pdf

22. M. Mohri, F. Pereira, and M. Riley, "Weighted finite-state transducers in speech recognition," *Computer Speech & Language*, vol. 16, no. 1, pp. 69–88, 2002. [Online]. Available: http://repository.upenn.edu/cgi/viewcontent.cgi?article=1010&context=cis_papers

23. R. Kneser and H. Ney, "Improved backing-off for m-gram language modeling," in *Acoustics, Speech, and Signal Processing, 1995. ICASSP-95., 1995 International Conference on*, vol. 1. IEEE, 1995, pp. 181–184. [Online]. Available: http://www-i6.informatik.rwth-aachen.de/publications/download/951/Kneser-ICASSP-1995.pdf

24. V. Panayotov, "Decoding graph construction in kaldi: A visual walkthrough," *Personal blog*, 2012. [Online]. Available: http://vpanayotov.blogspot.ru/2012/06/kaldi-decoding-graph-construction.html

25. M. Mohri, F. Pereira, and M. Riley, "Speech recognition with weighted finite-state transducers," in *Springer Handbook of Speech Processing*. Springer, 2008, pp. 559–584. [Online]. Available: http://www.cs.nyu.edu/~mohri/pub/hbka.pdf

26. Y. Bengio, R. Ducharme, P. Vincent, and C. Jauvin, "A neural probabilistic language model," *Journal of machine learning research*, vol. 3, no. Feb, pp. 1137–1155, 2003. [Online]. Available: http://www.jmlr.org/papers/volume3/bengio03a/bengio03a.pdf

27. T. Mikolov, M. Karafiát, L. Burget, J. Cernockỳ, and S. Khudanpur, "Recurrent neural network based language model." in *Interspeech*, vol. 2, 2010, p. 3. [Online]. Available: http://www.fit.vutbr.cz/research/groups/speech/servite/2010/rnnlm_mikolov.pdf

28. Y. Kim, Y. Jernite, D. Sontag, and A. M. Rush, "Character-aware neural language models," *arXiv preprint arXiv:1508.06615*, 2015. [Online]. Available: https://arxiv.org/pdf/1508.06615.pdf

29. M. Gutmann and A. Hyvärinen, "Noise-contrastive estimation: A new estimation principle for unnormalized statistical models." in *AISTATS*, vol. 1, no. 2, 2010, p. 6. [Online]. Available: http://www.jmlr.org/proceedings/papers/v9/gutmann10a/gutmann10a.pdf

30. S. J. Young, N. Russell, and J. Thornton, *Token passing: a simple conceptual model for connected speech recognition systems.* Cambridge University Engineering Department Cambridge, UK, 1989.

31. A. Hannun, "Sequence modeling with ctc," *Distill*, 2017, https://distill.pub/2017/ctc.

32. M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer, "Deep contextualized word representations," *arXiv preprint arXiv:1802.05365*, 2018.

33. A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, "Language models are unsupervised multitask learners," *OpenAI Blog*, vol. 1, no. 8, 2019.

34. J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

35. S. Kapadia, V. Valtchev, and S. J. Young, "Mmi training for continuous phoneme recognition on the timit database," in *1993 IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 2. IEEE, 1993, pp. 491–494.

36. B. Merialdo, "Phonetic recognition using hidden markov models and maximum mutual information training," in *ICASSP-88., International Conference on Acoustics, Speech, and Signal Processing.* IEEE, 1988, pp. 111–114.

37. D. Povey and P. C. Woodland, "Minimum phone error and i-smoothing for improved discriminative training," in *2002 IEEE International*

Conference on Acoustics, Speech, and Signal Processing, vol. 1. IEEE, 2002, pp. I–105.

38. K. Veselỳ, A. Ghoshal, L. Burget, and D. Povey, "Sequence-discriminative training of deep neural networks." in *Interspeech*, vol. 2013, 2013, pp. 2345–2349.

39. R. Prabhavalkar, T. N. Sainath, Y. Wu, P. Nguyen, Z. Chen, C.-C. Chiu, and A. Kannan, "Minimum word error rate training for attention-based sequence-to-sequence models," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018, pp. 4839–4843.

40. L. Wu, F. Tian, T. Qin, J. Lai, and T.-Y. Liu, "A study of reinforcement learning for neural machine translation," *arXiv preprint arXiv:1808.08866*, 2018.

41. A. Ratnaparkhi, "A maximum entropy model for part-of-speech tagging," in *Conference on Empirical Methods in Natural Language Processing*, 1996.

42. V. Valtchev, "Discriminative methods in hmm-based speech recognition," Ph.D. dissertation, University of Cambridge, 1995.

43. M. Shannon, "Optimizing expected word error rate via sampling for speech recognition," *arXiv preprint arXiv:1706.02776*, 2017.

44. A. Graves and N. Jaitly, "Towards end-to-end speech recognition with recurrent neural networks." in *ICML*, vol. 14, 2014, pp. 1764–1772. [Online]. Available: http://www.jmlr.org/proceedings/papers/v32/graves14.pdf

45. V. Zhukov, E. Golikov, and M. Kretov, "Differentiable lower bound for expected bleu score," *arXiv preprint arXiv:1712.04708*, 2017.