

Распознавание речи

Виктор Китов
victorkitov.github.io



Содержание

- 1 Представление звуковой информации
- 2 Listen-Attend-Spell
- 3 Connectionist Temporal Classification

Задачи обработки и генерации звука

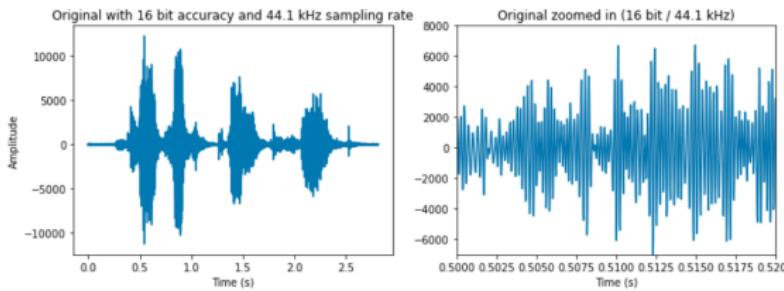
- Звук -> класс либо сегментация звуковой дорожки
 - голосовой помощник: команда / фоновый шум
 - определение композиции (shazam)
 - категоризация музыкального стиля (spotify)
- Звук -> сегментация
 - разметка спикеров
- Звук -> текст (automatic speech recognition, ASR)
- Текст -> звук (text to speech, TTS)
- Удаление шумов (denoising)
- Повышение качества аудио
 - ↑ частоты дискретизации (bandwidth expansion)
- Стилизация голоса
- Генерация музыки

Звук (waveform)¹

- Звук - последовательность импульсов звуковой волны

x_1, x_2, \dots, x_T - силы давления звуковой волны

- Так звук представлен в wav файле.

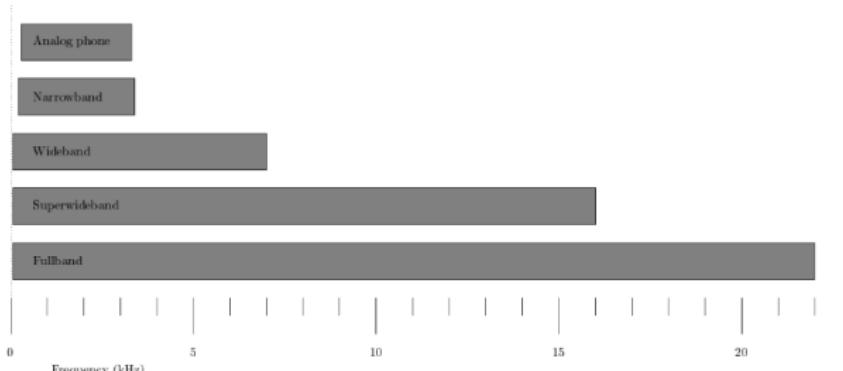


- Две характеристики качества:
 - частота (sampling frequency)-расстояние между t и $t + 1$
 - точность представления амплитуд x_t

¹Introduction to Speech Processing.

Частота (sampling frequency)

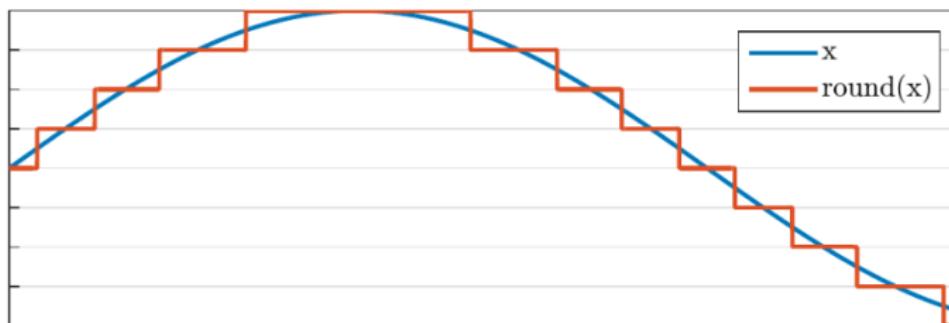
- Частота измеряется в Герцах (1 Регц (Hz)= 1 сек^{-1} - одно колебание в секунду)
- Частота 300-3500 кГц - большинство звуков речи
 - некоторые звуки (как "с") выше
- 16 кГц - достаточно в большинстве случаев
- 48 кГц - частота на компакт-дисках
 - высокие частоты нужны для не речевых сигналов, музыки



Квантизация сигнала - равномерная

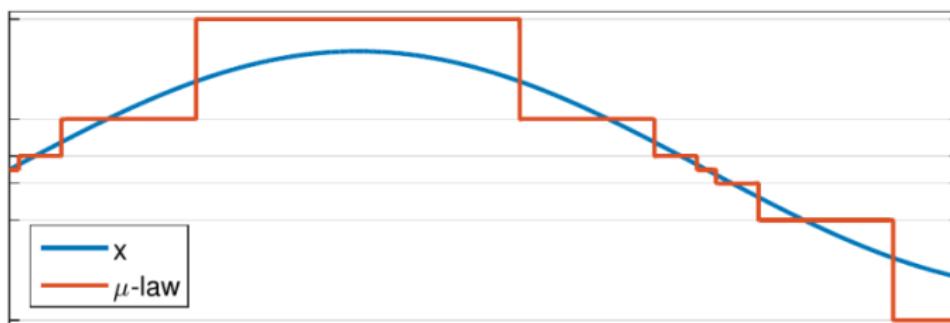
- На устройствах звук представляется в целочисленном виде (int), используя квантизацию.
- Равномерная квантизация:

$$\hat{x} = \Delta q \cdot \text{round}(x/\Delta q)$$



Квантизация сигнала - по μ -закону

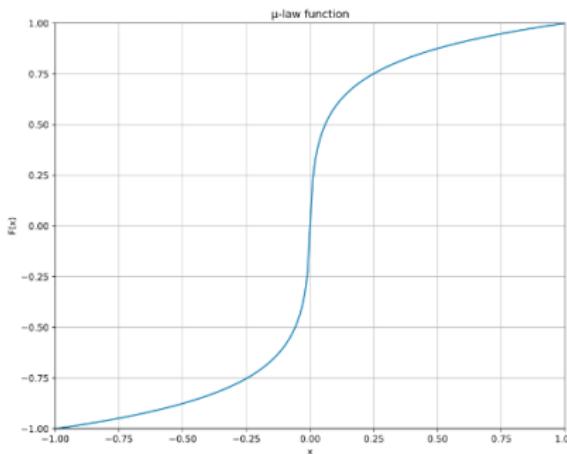
- Громкий звук выходит за интервал int-представления
- Человек силу звука воспринимает логарифмически.
 - выше чуткость тихих звуков, ниже - громких
- Поэтому квантизуют звук, преобразованный по μ -закону:



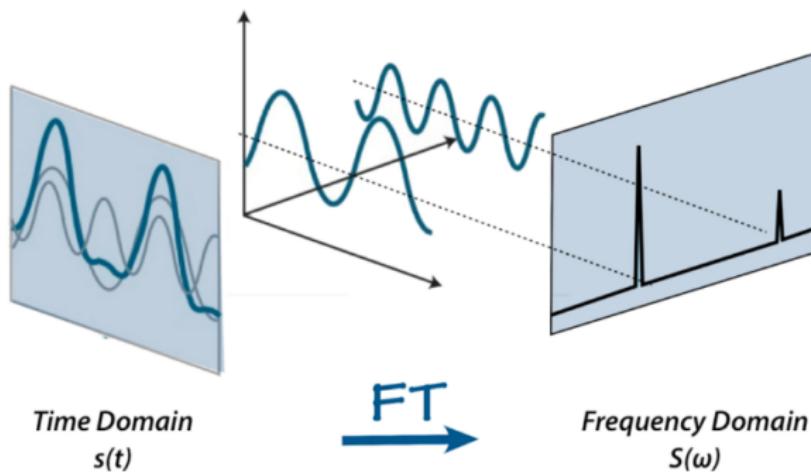
Квантизация сигнала - по μ -закону

Квантизуется сигнал, преобразованный по μ -закону:

$$x' = \text{sign}(x) \frac{\log(1 + \mu|x|)}{\log(1 + \mu)}$$



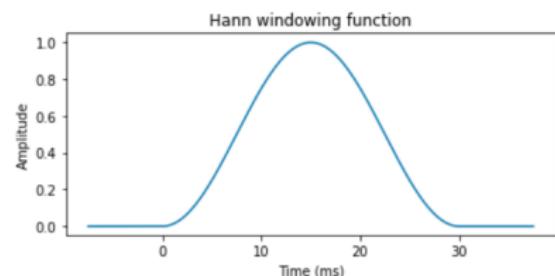
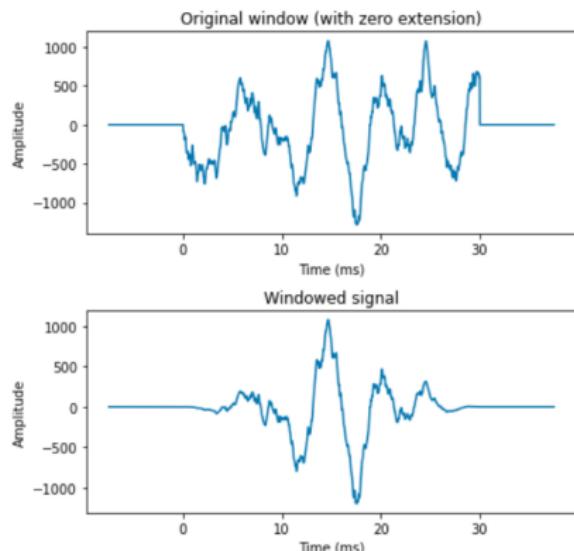
Представление звука²



²Ссылка на иллюстрацию.

Обработка каждого фрагмента звука

- Звук режется на пересекающиеся окна (длины $\sim 20\text{мс}$).
- Для удаления артефактов обрезки домножаем на оконную функцию:



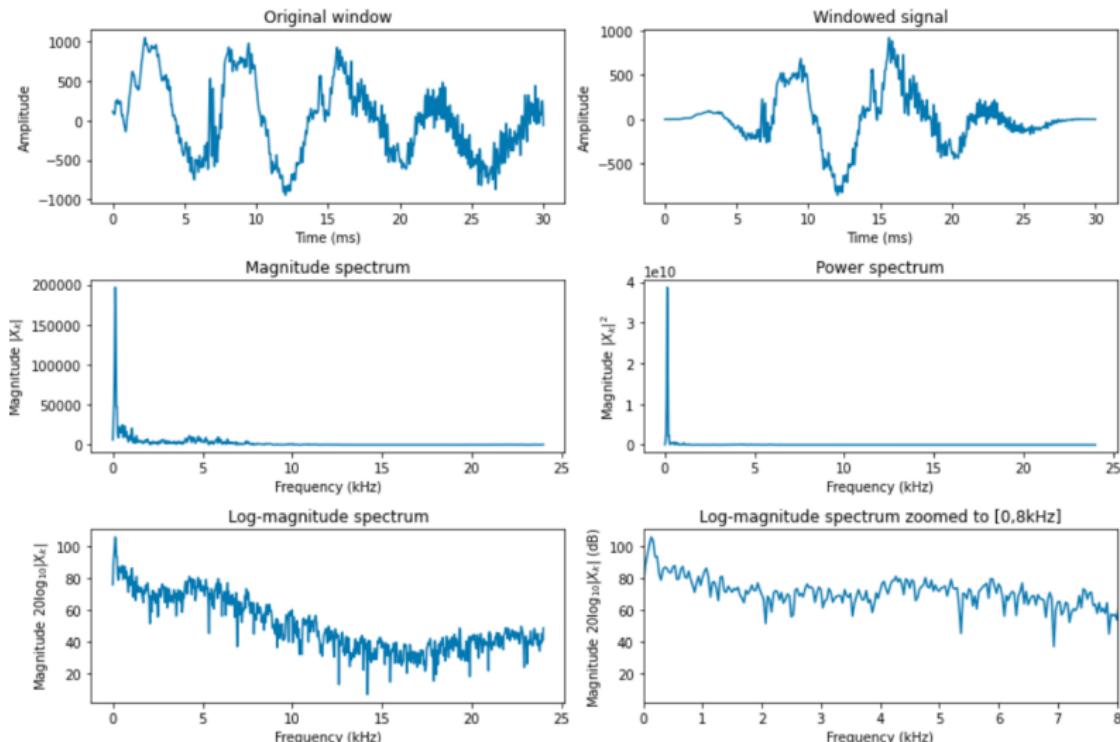
Спектр

- Для извлечения характеристик каждого фрагмента используется дискретное преобразование Фурье

$$X_k = \sum_{n=0}^{N-1} \tilde{x}_n e^{-i2\pi kn/N} = \sum_{n=0}^{N-1} x_n \cos(2\pi kn/N) - i \sum_{n=0}^{N-1} x_n \sin(2\pi kn/N)$$

- это коэффициенты разложения сигнала по \sin, \cos разной частоты
- X_k -комплексный, поэтому анализируют $|X_k|$ или $|X_k|^2$.
- Силы частот измеряют децибелах= $20 \log_{10} |X_k|$.
 - человек силу звука воспринимает логарифмически

Построение лог-спектрограммы



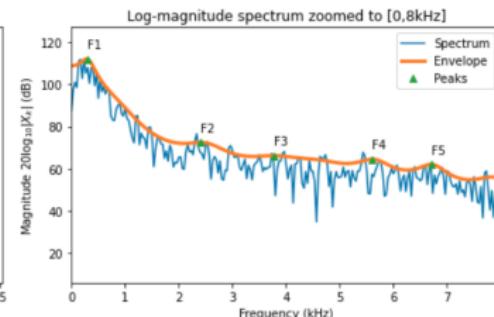
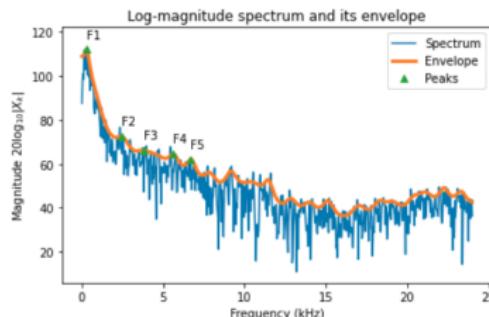
Пики на огибающей

Пики на огибающей спектра

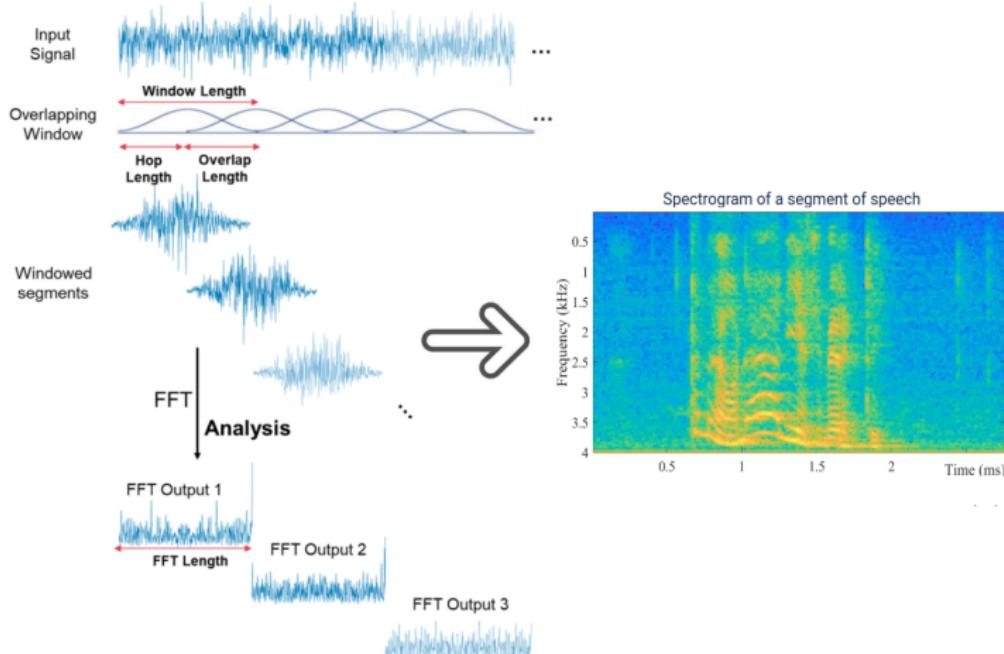
- называются формантами
- их частоты однозначно идентифицируют гласные звуки
- называются F_1, F_2, \dots

Частота колебания голосовых связок называется фундаментальной частотой F_0

- не связана с формантами и характеризует высоту голоса человека



Этапы построения спектрограммы³

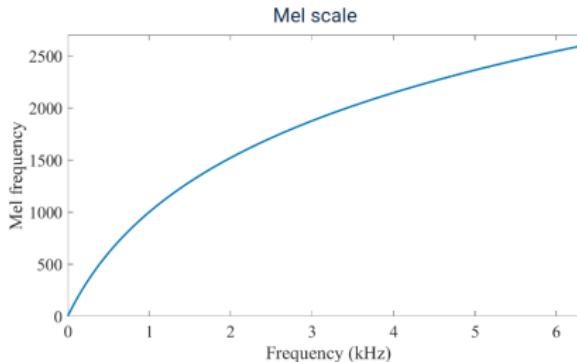


³[Ссылка на иллюстрацию.](#)

Мел-спектрограмма

- Человек воспринимает звук логарифмически.
 - ноты до на каждой след октаве - частота в 2 раза выше
- Эмпирический закон восприятия:

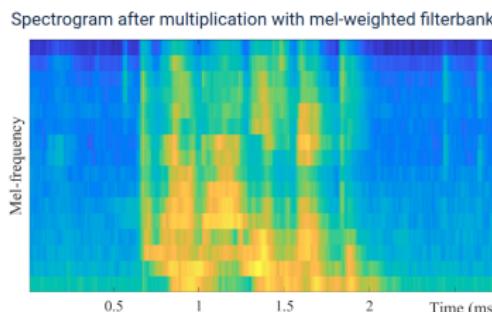
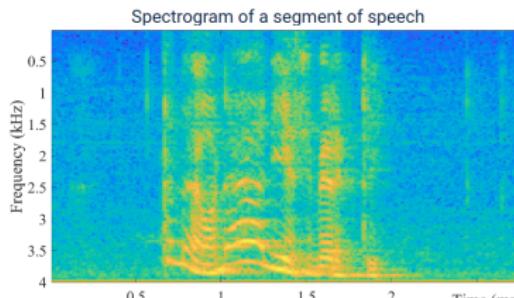
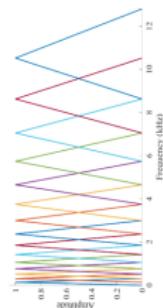
$$m = 2595 \log 10\left(1 + \frac{f}{700}\right)$$



- Также #частот избыточно, можно агрегировать соседние.

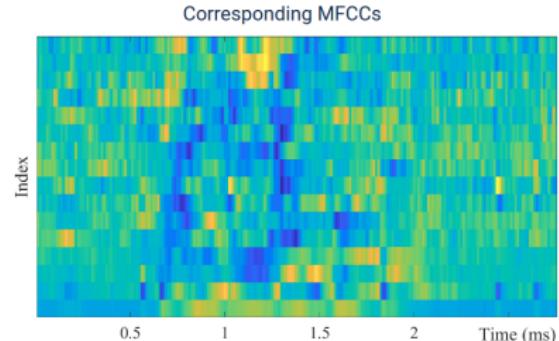
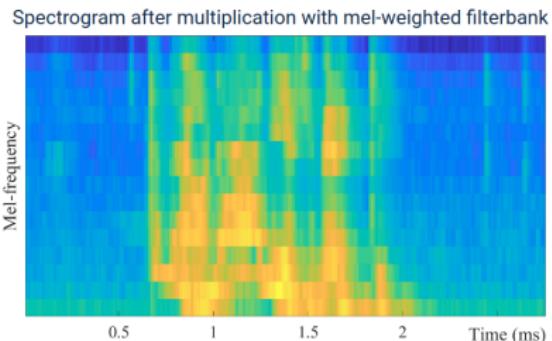
Мел-спектрограмма

Усредняем соседние частоты логарифма-спектрограммы по логарифмическому закону:



Кепстр, MFCC⁴

- Также в качестве признаков используют косинусное преобразование Фурье (вдоль частот)
 - к лог-спектограмме (кепстр, cepstr)
 - мел-спектограмме (mel-frequency cepstral coefficients, MFCCs).
- Получаем декоррелированные признаки, характеризующие частоты в целом и их огибающую.



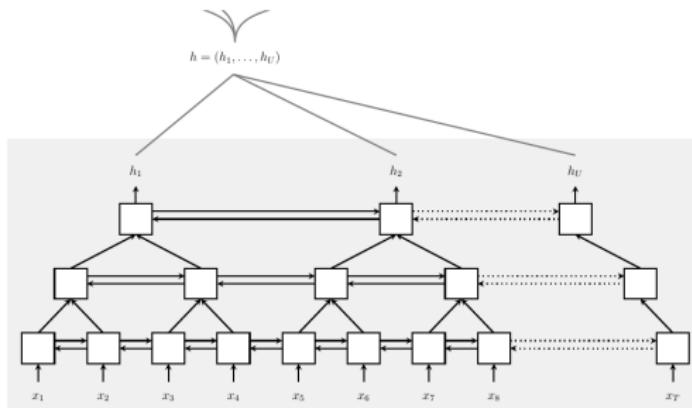
⁴<https://habr.com/ru/post/140828/>

Содержание

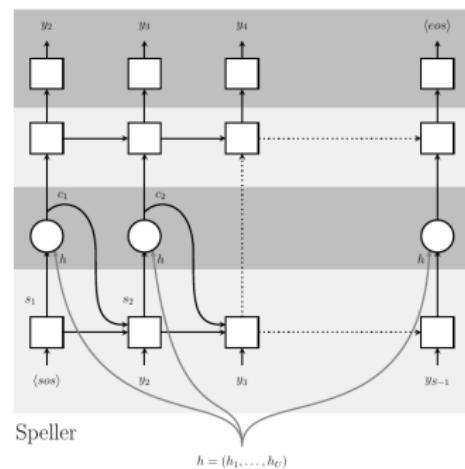
- 1 Представление звуковой информации
- 2 Listen-Attend-Spell
- 3 Connectionist Temporal Classification

Модель Listen-Attend-Spell⁵

- Listen-Attend-Spell - распознавание речи с помощью seq2seq+attention.
- Выход - распределение символов, начиная с `<sos>` и заканчивая `<eos>`.



Listener



Speller

⁵<https://arxiv.org/abs/1508.01211>

Модель Listen-Attend-Spell

- Декодер выдаёт распределение на символах
 - {a, b, c, . . . , z, 0, . . . , 9, <space>, <comma>, <period>, <apostrophe>, <unk> (для прочих символов).
- На вход декодеру (после <sos>)
 - с $p = 0.9$: реальный символ (teacher forcing)
 - с $p = 0.1$: ранее сгенерированный (free run)
 - учим модель исправлять ошибки
- Энкодер - 3x уровневый bidirectional LSTM (BLSTM)
- Перерасчет состояния на уровне I :
 - уменьшение в 2 раза длины последовательности

$$h_t^I = BLSTM \left(h_{t-1}^I, \left[h_{2t}^{I-1}, h_{2t+1}^{I-1} \right] \right)$$

- 3 уровня, длина выходных состояний в 2^3 раз короче длины входа.

Внимание в декодировщике

$$c_i = \text{AttentionContext}(s_i, h)$$

$$s_i = \text{RNN}(s_{i-1}, y_{i-1}, c_{i-1})$$

$$P(y_i | x, y_{<i}) = \text{CharacterDistribution}(s_i, c_i)$$

- Расчёт контекста c_i :

$$c_i = \sum_u \alpha_{i,u} h_u$$

$$\alpha_{i,u} = \frac{\exp(e_{i,u})}{\sum_u \exp(e_{i,u})}$$

$$e_{i,u} = \langle \phi(s_i), \psi(h_u) \rangle$$

- $\phi(\cdot), \psi(\cdot)$ - многослойные персептроны.

Генерация выходов

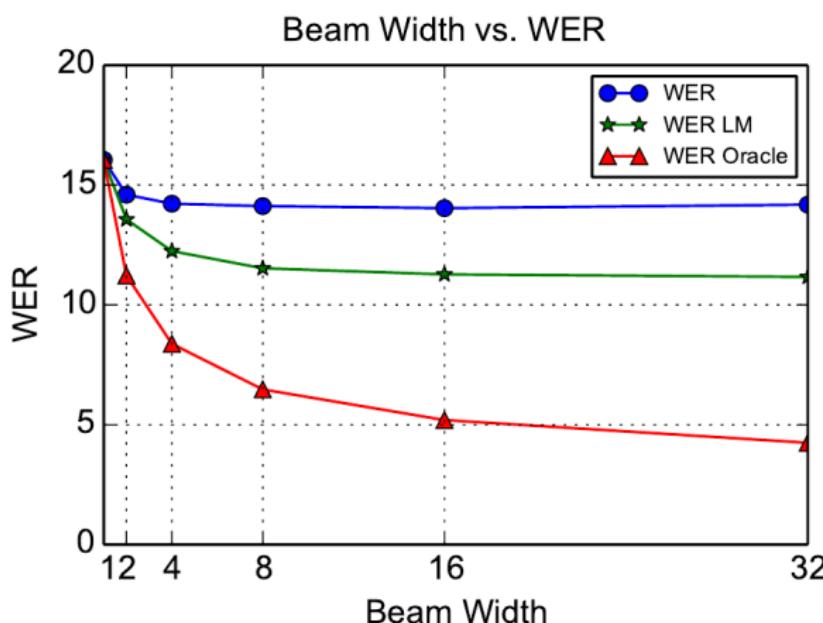
- Генерация выходной последовательности - через BeamSearch (32 лучших гипотезы)
- Считался score:

$$s(y|x) = \frac{\log P(y|x)}{|y|_c} + \lambda \log P_{LM}(y)$$

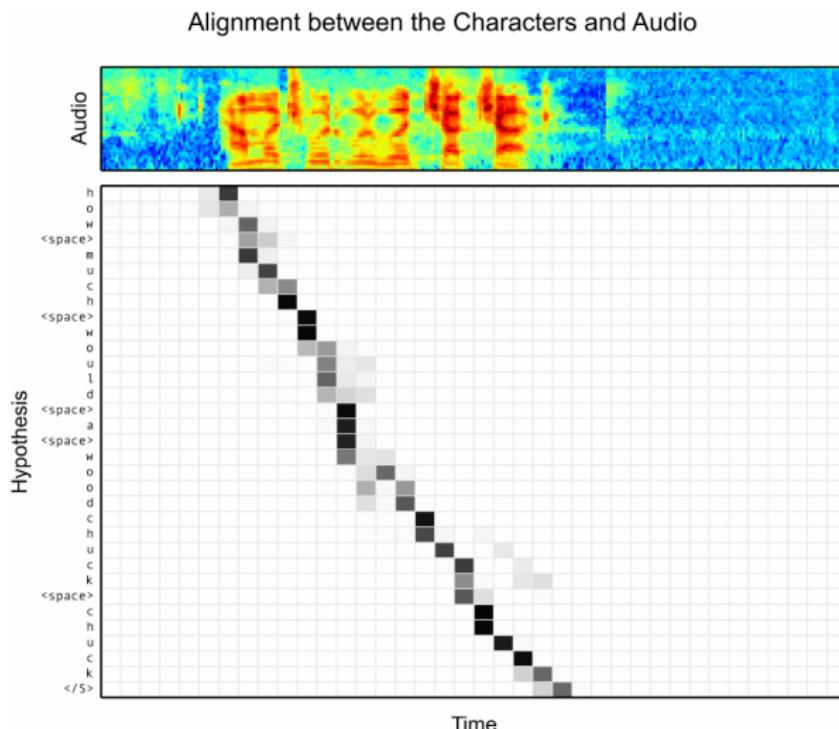
- $\log P < 0$, поэтому нормировка на #символов выхода $|y|_c$
↑ score
 - чтобы поощрить модель выдавать более длинные последовательности
- $P_{LM}(y)$ - вероятность у по языковой модели.
 - много текстов для обучения
 - существенно ↑ качество
- Аугментация при обучении:
 - добавление эхо (reverberations)
 - добавление внешних шумов (из видео YouTube)

Word-error-rate от #гипотез лучевого поиска

- С 16 гипотез качество почти не улучшается.
- Включение языковой модели ↑ качество



Визуализация внимания

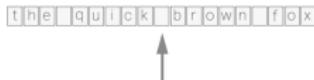


Содержание

- 1 Представление звуковой информации
- 2 Listen-Attend-Spell
- 3 Connectionist Temporal Classification
 - Агрегация по выравниваниям
 - Построение прогноза

CTC^{6,7}

- Требуется построить $x_1x_2\dots x_T \rightarrow y_1y_2\dots y_U$,
 - $T \geq U$, объекты посл-тей монотонно связаны во времени
- Примеры:
 - рукописный текст->текст
 - звук->текст
 - видео->разметка событий на фреймах



The quick brown fox

Handwriting recognition: The input can be (x, y) coordinates of a pen stroke or pixels in an image.



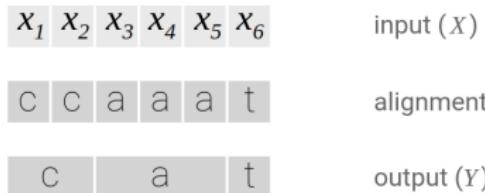
Speech recognition: The input can be a spectrogram or some other frequency based feature extractor.

⁶Идея метода.

⁷Туториал и иллюстрации по СТС.

Звук->токен->символ: $x_t \rightarrow a_t \xrightarrow{?} y_s$

- Кодировщик звука: звук в токен $x_t \rightarrow a_t$ (выдаёт $p(a_t|x_t)$)
 - технически: сначала CNN (conv1d или широкая свёртка), потом RNN.



- Many-to-one: $a_1 a_2 \dots a_T \rightarrow y_1 y_2 \dots y_s, s \leq T$
- Проблема: не знаем выравнивания между X/A и $Y!$
 - какой звук произносился долго/коротко
 - могут возникать паузы

Проблема агрегации по выравниваниям

- Необходимо оценивать и максимизировать $p(Y|X)$:
 - для настройки модели по обучающим (X, Y)
 - для распознавания речи X в новых данных

$$p(Y | X) = \sum_{A \in \mathcal{A}_{X,Y}} \prod_{t=1}^T p_t(a_t | X)$$

The CTC conditional probability

marginalizes over the set of valid alignments

computing the **probability** for a single alignment step-by-step.

- Агрегировать по всем выравниваниям - долго.
- Алгоритм СТС агрегирует по всем возможным выравниваниям токенов $A = [a_1 a_2 \dots a_T]$ за полиномиальное время.

СТС преобразование: $A \rightarrow Y$

- Введем пустой символ ϵ в кодировке a_t .
- СТС преобразование: $A \rightarrow Y$:
 - ① объединить повторяющиеся символы в один
 - звук в речи может тянуться
 - ② убрать ϵ (за счёт этого символы в Y могут повторяться)
 - введение ϵ позволяет выводить повторяющиеся символы, например, hello

h h e ε ε | | | ε | | o

First, merge repeat characters.

h e ε | ε | o

Then, remove any ϵ tokens.

h e | | o

The remaining characters are the output.

h e | | o

Выравнивание и кодировка

Valid Alignments

$\epsilon \ C \ C \ \epsilon \ a \ t$

$c \ c \ a \ a \ t \ t$

$c \ a \ \epsilon \ \epsilon \ \epsilon \ t$

Invalid Alignments

$C \ \epsilon \ C \ \epsilon \ a \ t$

$c \ c \ a \ a \ t \ _$

$C \ \epsilon \ \epsilon \ \epsilon \ | t \ t$

corresponds to
 $Y = [c, c, a, t]$

has length 5

missing the 'a'

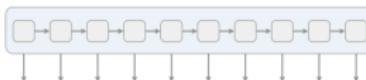
- Свойства СТС преобразования:

- монотонность соответствий символов $A_T \rightarrow Y_s$
- отображение $A_T \rightarrow Y_s$ many-to-one, $|A| \geq |Y|$

Последовательность



We start with an input sequence,
 like a spectrogram of audio.



The input is fed into an RNN,
 for example.

h	h	h	h	h	h	h	h	h	h
e	e	e	e	e	e	e	e	e	e
o	o	o	o	o	o	o	o	o	o
€	€	€	€	€	€	€	€	€	€

The network gives $p_t(a | X)$,
 a distribution over the outputs
 $\{h, e, |, o, \epsilon\}$ for each input step.

h	e	€			€			o	o
h	h	e			€	€		€	o
€	e	€			€	€		o	o

With the per time-step output
 distribution, we compute the
 probability of different sequences

h	e	l	l	o
e	l	l	o	
h	e	l	o	

$$p(Y | X) = \sum_{A \in \mathcal{A}_{X,Y}} \prod_{t=1}^T p_t(a_t | X)$$

The CTC conditional
 probability

marginalizes over the
 set of valid alignments

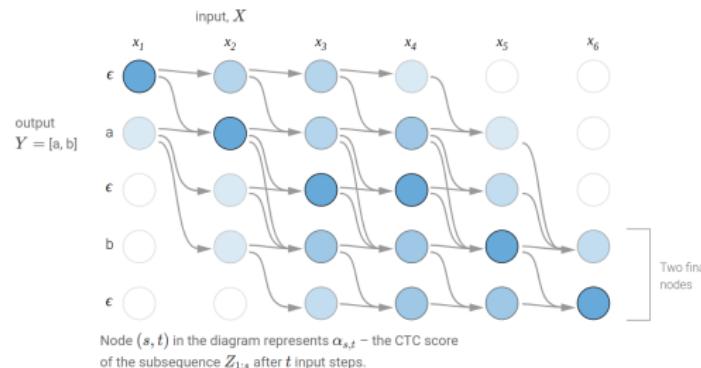
computing the probability for
 single alignment step-by-step.

By marginalizing over alignments,
 we get a distribution over outputs.

3 Connectionist Temporal Classification

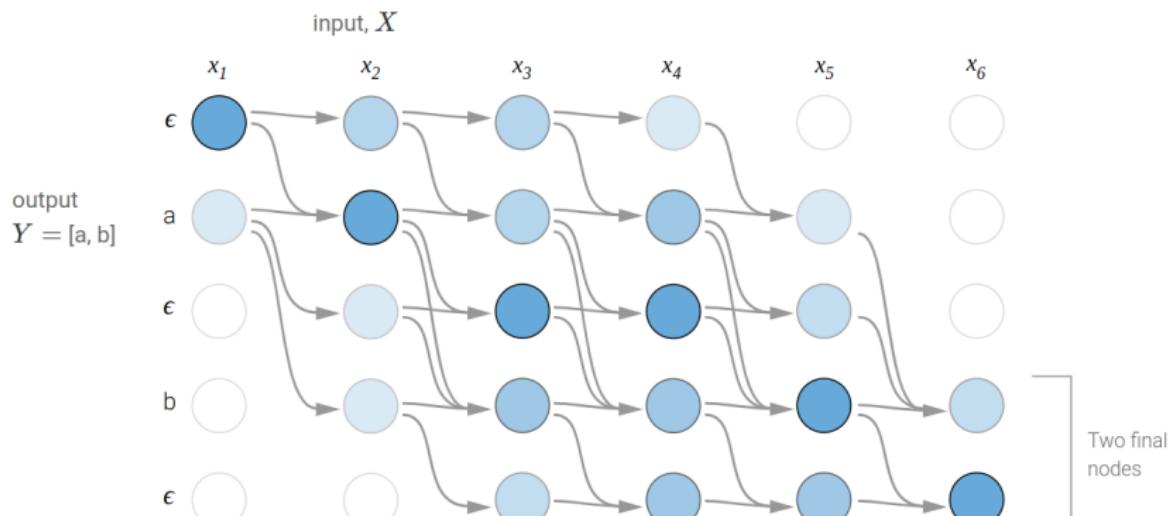
- Агрегация по выравниваниям
- Построение прогноза

Агрегация по выравниваниям



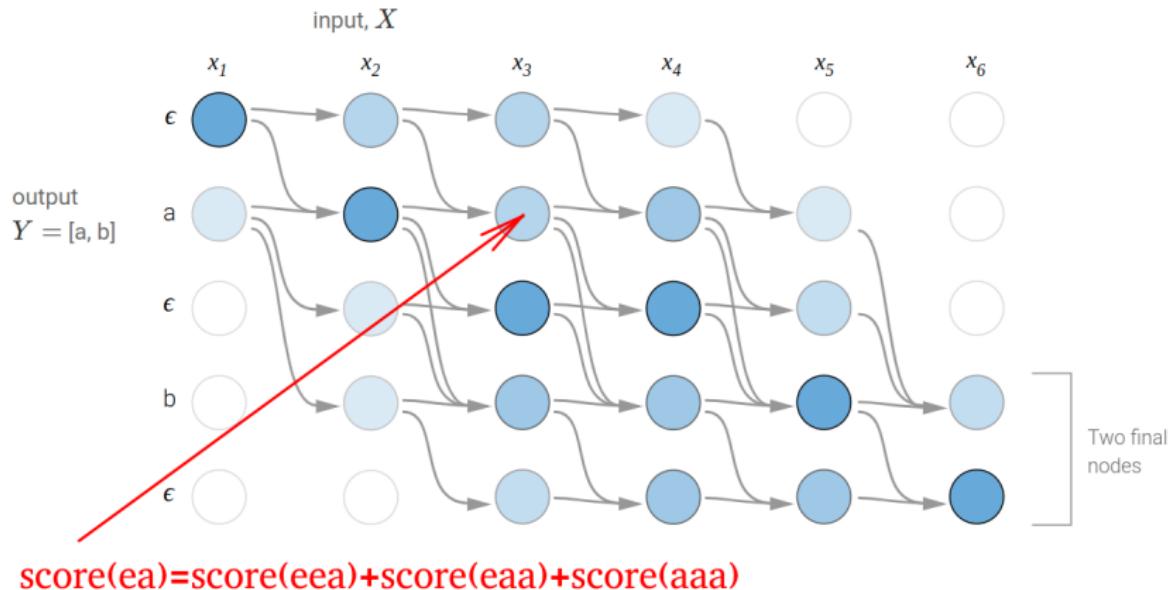
- Обучающий объект $(x_1 x_2 \dots x_T; y_1 y_2 \dots y_N)$, $P(Y|X) - ?$
- $Y = [y_1 y_2, \dots y_N] \rightarrow Z = [\varepsilon y_1 \varepsilon y_2 \varepsilon \dots \varepsilon y_N \varepsilon]$ - по строкам,
 - пример: $Y = 'hello' \rightarrow Z = '\varepsilon h \varepsilon e \varepsilon e l \varepsilon l \varepsilon o \varepsilon'$
- По столбцам- $t = 1, 2, \dots T$, по строкам Y .
- Инициализация: $\alpha_{u1} = p(z_s | x_1)$
- Далее рекуррентно считаем $\alpha_{ut} = P(Z_{:u} | X_{:t})$ для $u, t \uparrow$.
- Итоговый $P(y_1 \dots y_N | x_1 \dots x_T) = \alpha_{|Z|-1, T} + \alpha_{|Z|, T}$

Агрегация по выравниваниям

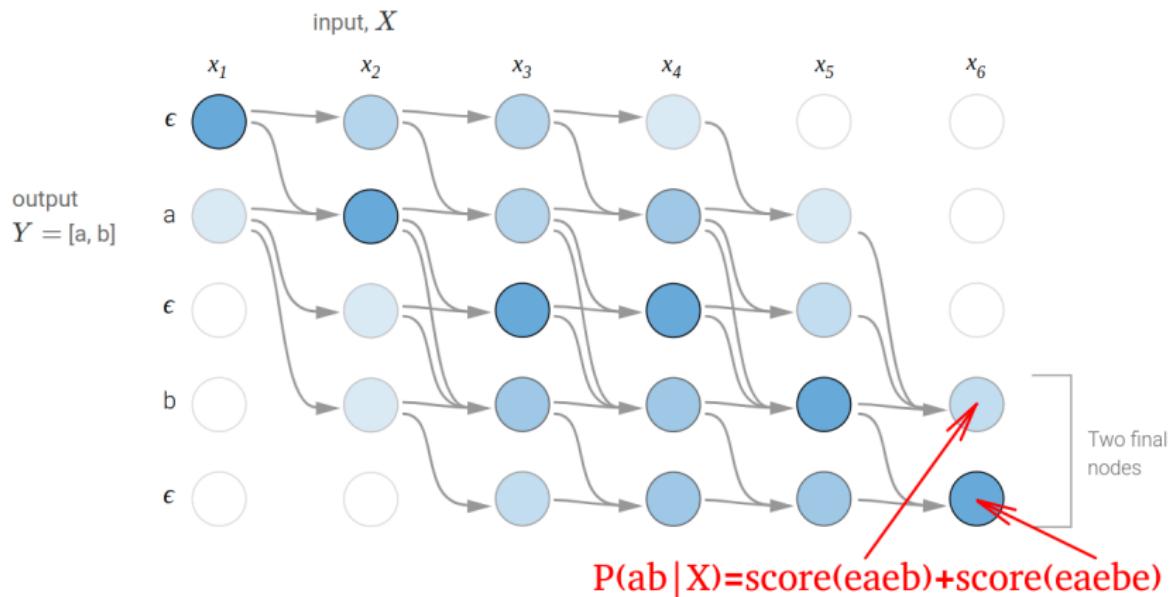


Node (s, t) in the diagram represents $\alpha_{s,t}$ – the CTC score of the subsequence $Z_{1:s}$ after t input steps.

Агрегация по выравниваниям: промежуточный узел



Агрегация по выравниваниям: итоговый $P(Y|X)$

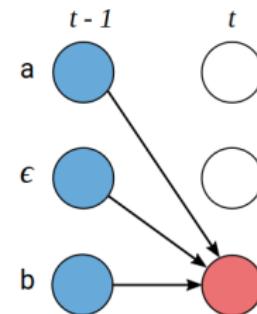


Варианты агрегации

- СТС-кодирование: $Y = [y_1 y_2, \dots y_N] \rightarrow Z = [\varepsilon y_1 \varepsilon y_2 \varepsilon \dots \varepsilon y_N \varepsilon]$
 - пример: $Y = 'hello' \rightarrow Z = ' \varepsilon h \varepsilon e \varepsilon e \varepsilon l \varepsilon l \varepsilon o \varepsilon '$
 - Z идёт по строкам
- Нужно уметь обрабатывать случаи:
 - $a\varepsilon b$ (случай 1)
 - $a\varepsilon a$ (случай 2)
 - $\varepsilon a\varepsilon$ (случай 3)
- Случаи $ab\varepsilon$, εab невозможны из-за вида СТС-кодирования
 - т.к. разделяем символы ε .

Случай 1 (сумма 3х предыдущих состояний)

In the second case, we're allowed to skip the previous token in Z . We have this case whenever z_{s-1} is an ϵ between unique characters. As a result there are three positions we could have come from at the previous step.

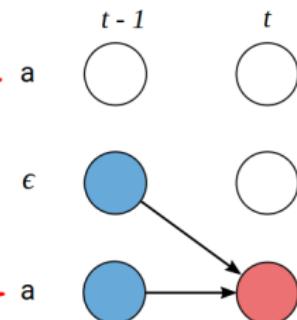


$$\alpha_{s,t} = (\alpha_{s-2,t-1} + \alpha_{s-1,t-1} + \alpha_{s,t-1}) \cdot \underbrace{p(z_s|X)}_{\text{из модели для выхода } t}$$

Случай 2 (сумма 2x предыдущих состояний)

In this case, we can't jump over z_{s-1} , the previous token in Z .

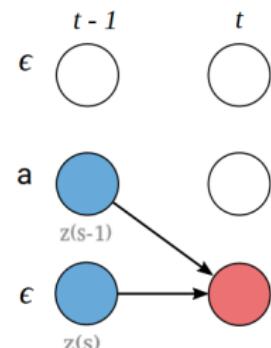
The second reason is that we must have an ϵ between repeat characters in Y . We can identify this when $z_s = z_{s-2}$.



$$\alpha_{s,t} = (\alpha_{s-1,t-1} + \alpha_{s,t-1}) \cdot \underbrace{p(z_s|X)}_{\text{из модели для выхода } t}$$

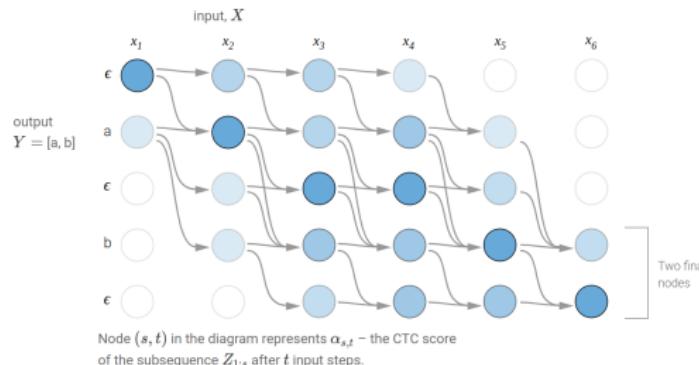
Случай 3 (сумма 2x предыдущих состояний)

In this case, we can't jump over z_{s-1} , the previous token in Z . The first reason is that the previous token can be an element of Y , and we can't skip elements of Y . Since every element of Y in Z is followed by an ϵ , we can identify this when $z_s = \epsilon$.



$$\alpha_{s,t} = (\alpha_{s-1,t-1} + \alpha_{s,t-1}) \cdot \underbrace{p(z_s|X)}_{\text{из модели для выхода } t}$$

Пример выравнивания



- Расчёт $P(ab|X)$: нужно агрегировать по 2м узлам на старте (ϵ и a) и 2м узлам на финише (b, ϵ).
- Можем эффективно вычислить $p(Y|X)$ и настраивать модель $f_\theta : X \rightarrow p(a|X)$ из

$$\sum_{(X, Y) \in TrainSet} \log p_\theta(Y|X) \rightarrow \max_\theta$$

3 Connectionist Temporal Classification

- Агрегация по выравниваниям
- Построение прогноза

Построение прогноза (наивный подход)

Построение прогноза

$$\hat{Y} = \arg \max_Y p(Y|X)$$

Наивный прогноз:

$$1) \hat{A} = \arg \max_A \prod_{t=1}^T p(a_t|X)$$

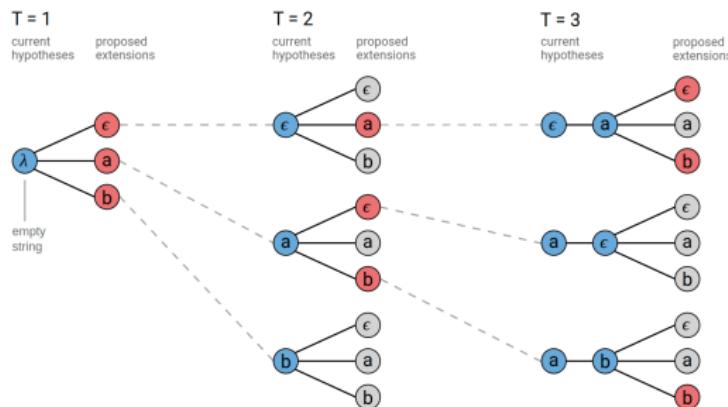
$$2) \hat{A} \rightarrow \hat{Y} \text{ через СТС преобразование}$$

Работает неточно, поскольку не учитывает, что одному Y могут соответствовать разные выравнивания.

- пример: $P(bb|X) > \max\{P(aa), P(a\varepsilon), P(\varepsilon a)\}$, но может быть $P(aa|X) + P(a\varepsilon|X) + P(\varepsilon a|X) > P(bb|X)$, поэтому правильно предсказывать "a", а не "b"!

Построение прогноза (наивный подход)

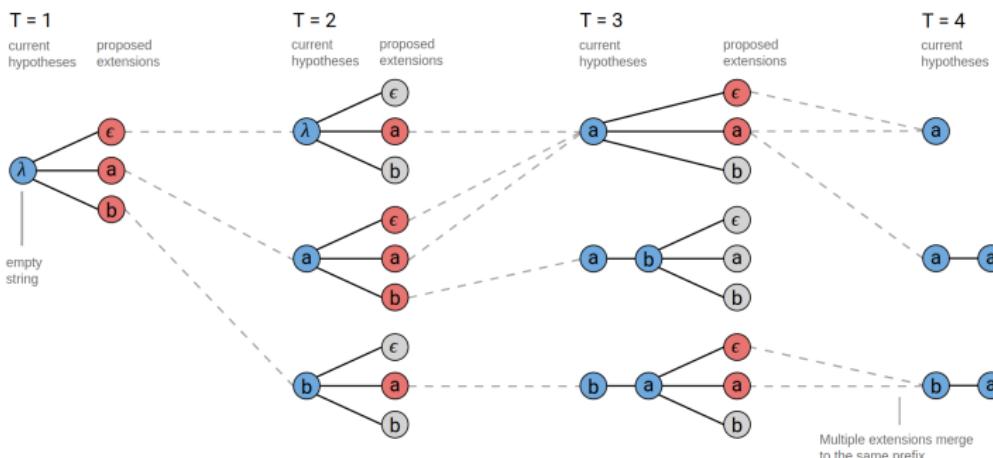
Иллюстрация лучевого поиска в наивном подходе:



A standard beam search algorithm with an alphabet of $\{\epsilon, a, b\}$ and a beam size of three.

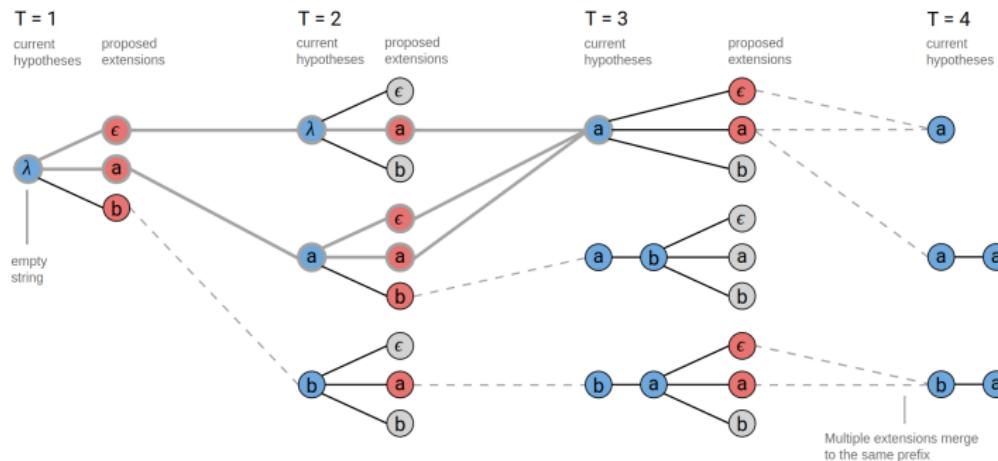
Лучевой поиск с учетом Y

Правильнее в лучевом поиске ранжировать не лучшие $A_{:t}$ гипотезы, а лучшие соответствующие $Y_{:s}$ гипотезы



Лучевой поиск с учетом Y (иллюстрация для узла)

Правильнее в лучевом поиске ранжировать не лучшие $A_{:t}$ гипотезы, а лучшие соответствующие $Y_{:s}$ гипотезы



- Для этого агрегируем

$$\epsilon a + a\epsilon + aa \rightarrow a, \epsilon aa + \epsilon a\epsilon \rightarrow a, baa + ba\epsilon \rightarrow ba, \dots$$

Построение прогноза с учетом выходного Y

- $\epsilon a + a\epsilon + aa \rightarrow a$ но внутри нужно разделять 2 случая (и запоминать их вероятности отдельно) $\epsilon a + aa \rightarrow a$ и $a\epsilon \rightarrow a$ т.к. при последующей склейке с a результат различный:

$$\epsilon a + a \rightarrow a, \text{ но } a\epsilon a \rightarrow aa$$

- При склейке же с b результат одинаковый:

$$\epsilon a + b \rightarrow ab \text{ и } a\epsilon b \rightarrow ab$$

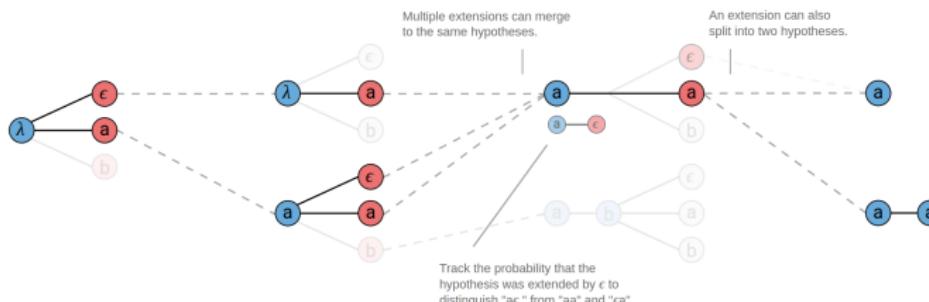
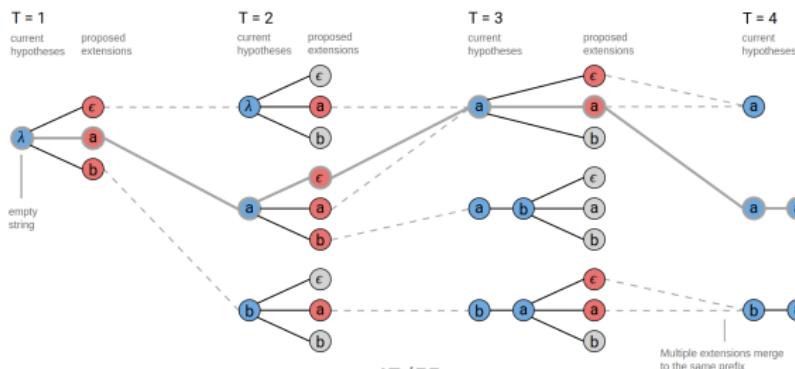
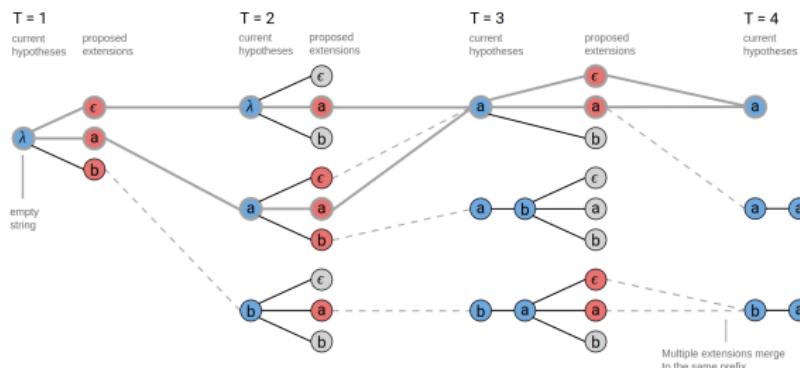


Иллюстрация разделения префиксов ... $a\varepsilon$ и ... a :



Повышение качества прогнозов

Улучшенный рейтинг для \uparrow качества выходов:

$$\text{лучевой поиск: } p(Y|X) \rightarrow p(Y|X) \cdot p(Y)^\alpha \cdot |Y|^\beta$$

- $p(Y)$ - языковая модель (можем оценить по большим корпусам текстов)
- $|Y|$ - длина последовательности (иначе поощряются более короткие из-за $p \times p \times p \dots$)

Комментарии

Проверка корректности СТС для обучающей пары (X, Y) :

- ① считаем $p_{true}(Y|X)$ по всем разбиениям
- ② считаем $p_{CTC,\beta}(Y|X)$ с помощью СТС и лучевого поиска ширины β

Должно получиться:

$$p_{CTC,\beta}(Y|X) \leq p_{true}(Y|X)$$
$$p_{CTC,\beta}(Y|X) \rightarrow p_{true}(Y|X) \text{ при } \uparrow \beta$$

Настройка ширины лучевого поиска β :

- β управляет противоречием вычислительная сложность-точность.
- берём минимальное β , когда $p_{CTC,\beta}(Y|X) \approx p_{true}(Y|X)$

Сравнение LAS и СТС моделей

- Преимущества СТС:

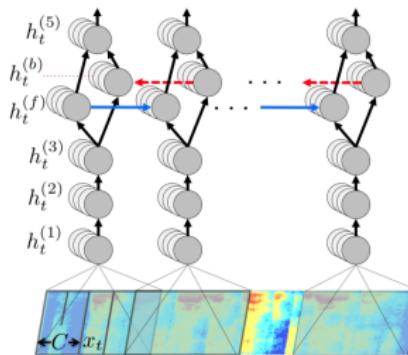
- не нужно информации о выравнивании звуков и соответствующих фраз
- проще обучение: нужно настраивать только кодировщик, без декодировщика

- Недостаток СТС:

- выходы независимы от предыдущих выходов: $p(y_t|X)$ не зависит от $y_{t-1}y_{t-2}, \dots$
 - начало распознанной фразы по идеи предопределяет её продолжение!
 - модель LAS это умеет за счёт рекуррентного декодировщика.
- Языковая модель отчасти исправляет этот недостаток СТС.

DeepSpeech⁸

- Модель DeepSpeech реализует СТС.
- X_t - окно спектрограммы ($\pm 3,5,7$ фреймов).
- Для прогнозирования $p(a_t|X_t)$ используются
 - 3 FC, left-to-right и right-to-left RNN, 1 FC (от суммы состояний RNN), SoftMax:



⁸<https://arxiv.org/pdf/1412.5567.pdf>

DeepSpeech - особенности

- FC слои можно воспринимать как 1D свёртку
 - т.к. веса FC слоев не зависят от t
- Использовался DropOut на всех FC слоях
 - $p_{drop} = 0.05, 0.1.$
 - но не к пересчету состояний RNN
- Во время обучения и теста использовался ансамбль:

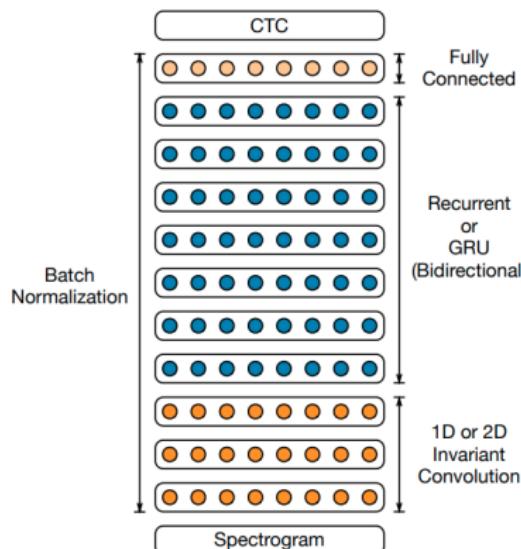
$$\frac{1}{3} (F(x_{-5ms}) + F(x) + F(x_{+5ms}))$$

- x_{Kms} - сигнал x , сдвинутый на K миллисекунд.
- Нелинейность - Clipped ReLU:

$$\text{ReLU}_{clip}(x) = \min \{\max \{x, 0\}, 20\}$$

DeepSpeech 2⁹

- DeepSpeech 2 также реализует СТС.
- Прогноз $p(a_t|X_t)$:



⁹<https://arxiv.org/pdf/1512.02595.pdf>

DeepSpeech 2 - детали архитектуры

- свёрточные слои (2D-conv по времени и частоте лучше себя показала)
- $\text{stride} > 1$ для \downarrow #параметров и вычислений
- 7 двунаправленных RNN (GRU)
- BatchNorm на всех слоях \uparrow качество прогнозов.
 - в RNN он использовался только при учёте нижестоящего слоя:

$$h_t^l = f \left(\text{BatchNorm} \left(Wh_t^{l-1} \right) + Uh_{t-1}^l \right)$$

Заключение

- Распознавание речи основано на
 - seq2seq+attention: Listen-Attend-Spell (LAS)
 - потери СТС: DeepSpeech, DeepSpeech 2.
- качество ↑ , если
 - используем языковую модель
 - поощряем длительность у
 - используем аугментацию
 - добавляется эхо
 - добавляем посторонний шум
 - учимся на x чуть смещенных по t