Распознавание речи

Виктор Китов

v.v.kitov@yandex.ru

Содержание

Задачи обработки и генерации звука

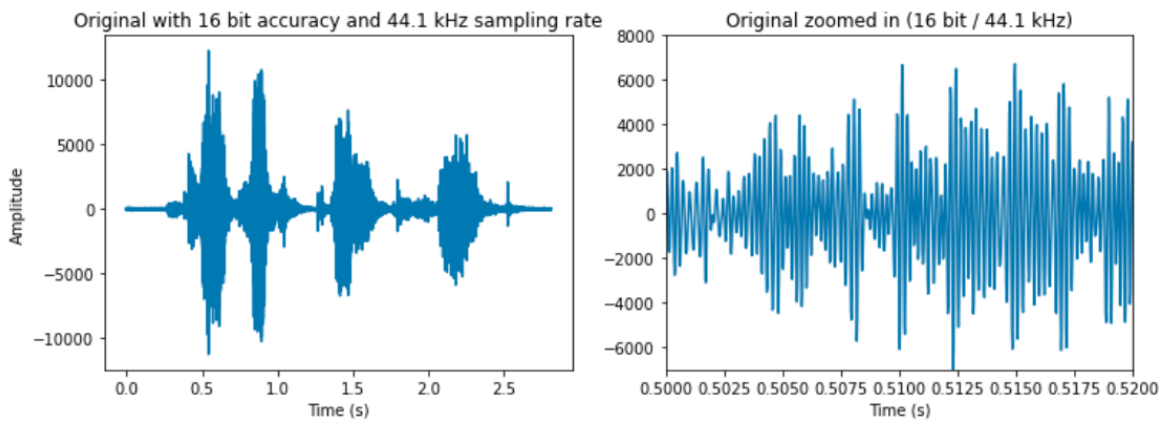
- Звук -> класс либо сегментация звуковой дорожки
  - голосовой помощник: команда / фоновый шум
  - определение композиции (shazam)
  - категоризация музыкального стиля (spotify)
- Звук -> сегментация
  - разметка спикеров
- Звук -> текст (automatic speech recognition, ASR)
- Текст -> звук (text to speech,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

Звук (waveform)[1]

- Звук - последовательность импульсов звуковой волны

  $x_1, x_2, ...x_T$ - силы давления звуковой волны

- Так звук представлен в wav файле.

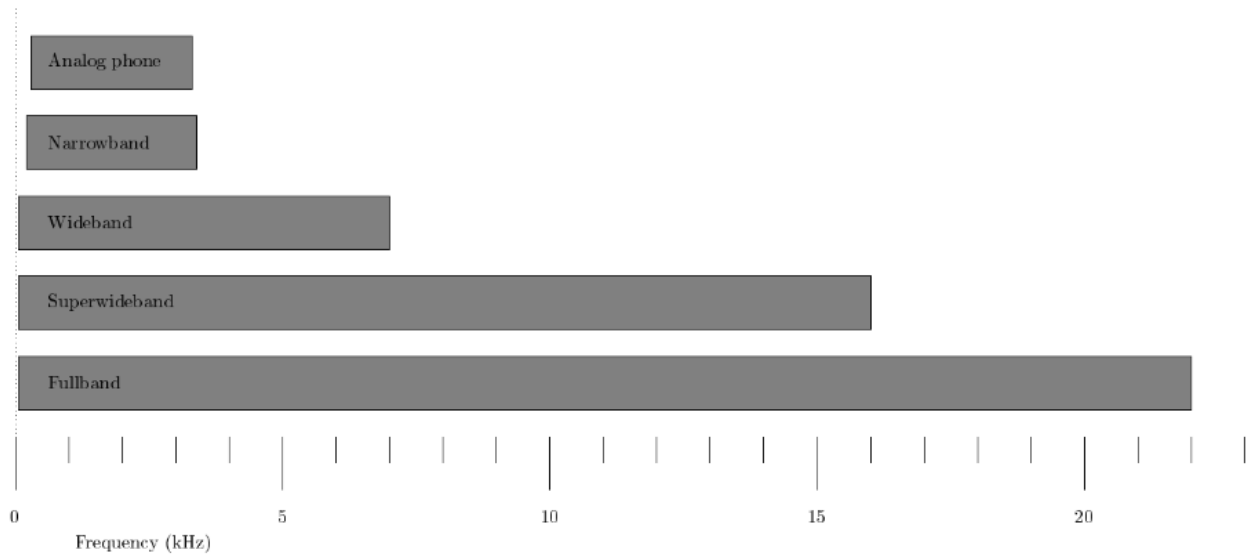


- Две характеристики качества:
  - частота (sampling frequency)-расстояние между t и $t+1$
  - точность представления амплитуд x_t

[1] Introduction to Speech Processing.

Частота (sampling frequency)

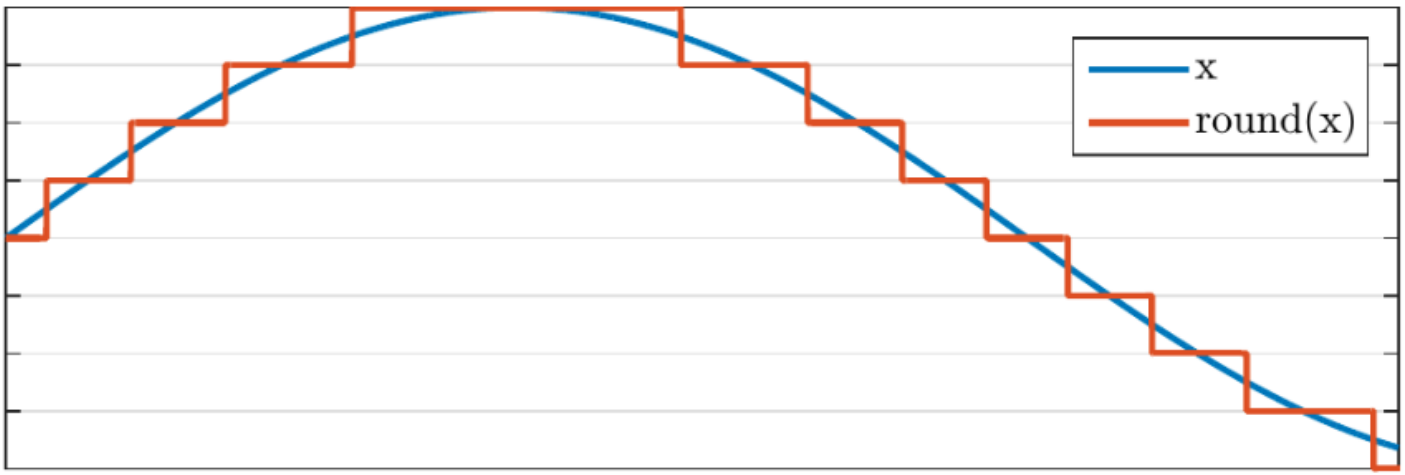
- Частота измеряется в Герцах (1 Регц (Hz)=1 сек$^{-1}$ - одно колебание в секунду)
- Частота 300-3500 кГц - ольшинство звуков речи
  - некоторые звуки (как "с") выше
- 16 кГц - достаточно в большинстве случаев
- 48 кГц - частота на компакт-дисках
  - высокие частоты нужны для не речевых сигналов, музыки



Квантизация сигнала - равномерная

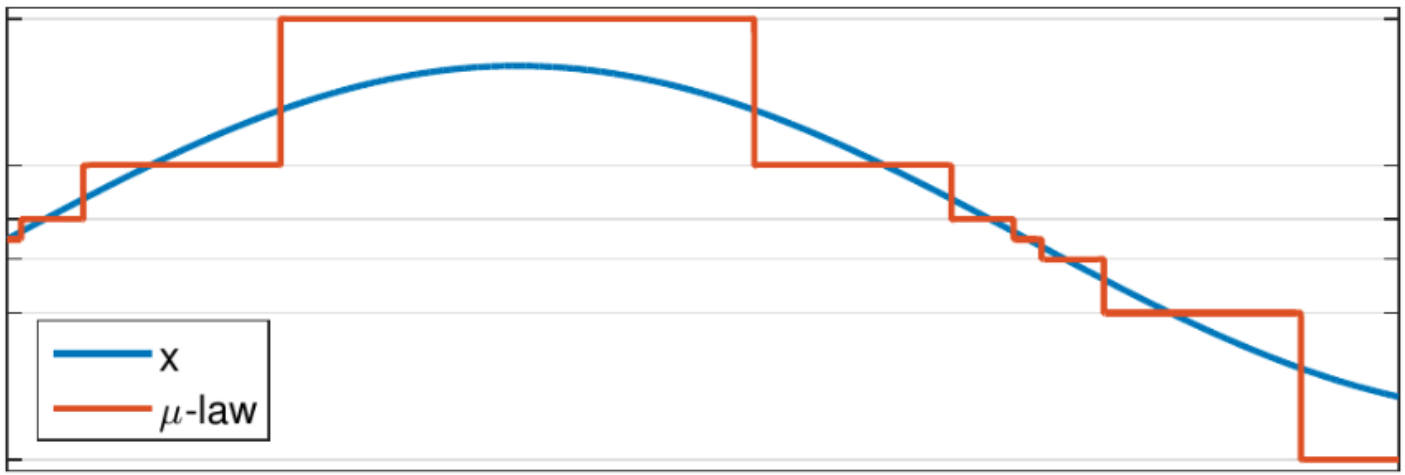
- На устройствах звук представляется в целочисленном виде (int), используя квантизацию.
- Равномерная квантизация:

$$\widehat{x} = \Delta q \cdot \text{round}\left(x/\Delta q\right)$$



Квантизация сигнала - по μ-закону

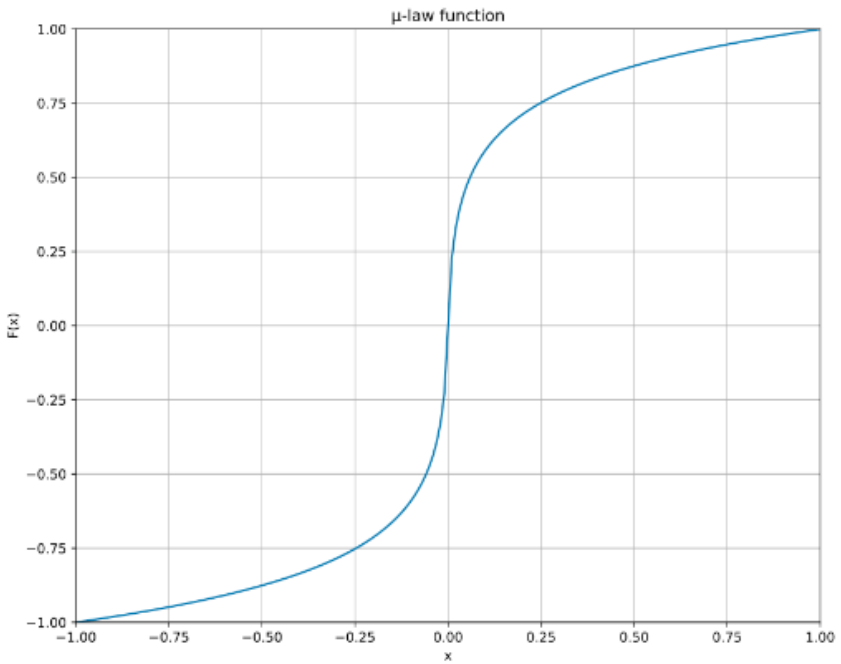
- Громкий звук выходит за интервал int-представления
- Человек силу звука воспринимает логарифмически.
  - выше чуткость тихих звуков, ниже - громких
- Поэтому квантизуют звук, преобразованный по μ-закону:



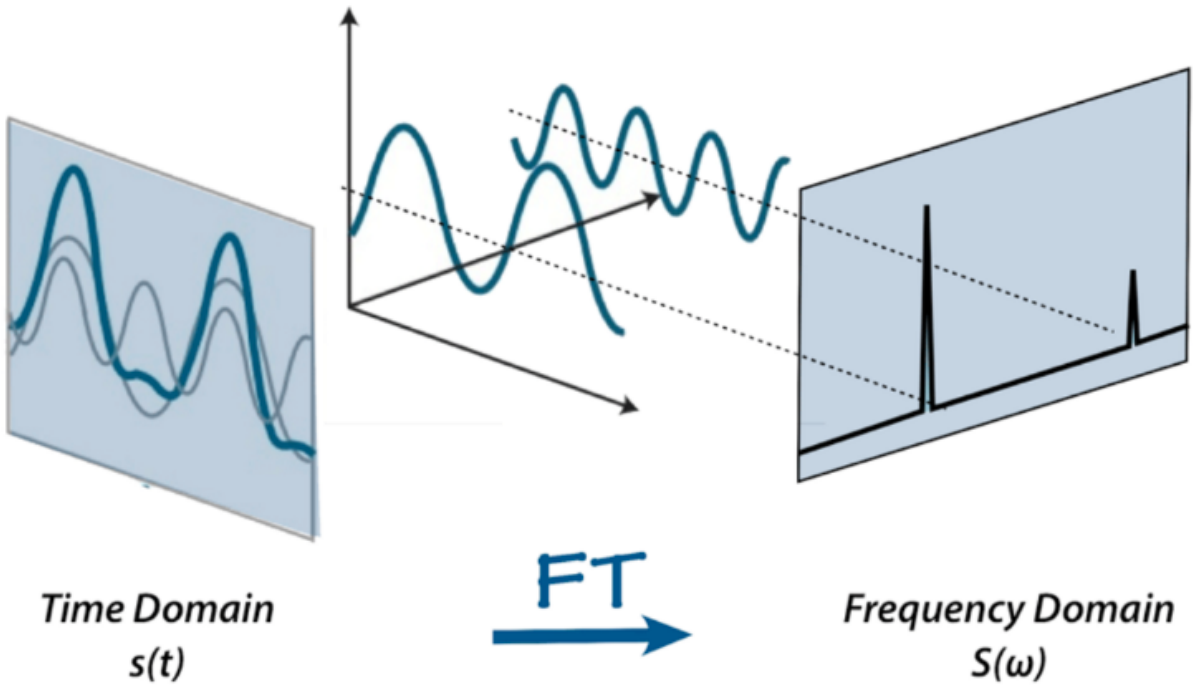
Квантизация сигнала - по μ-закону

Квантизуется сигнал, преобразованный по μ-закону:

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



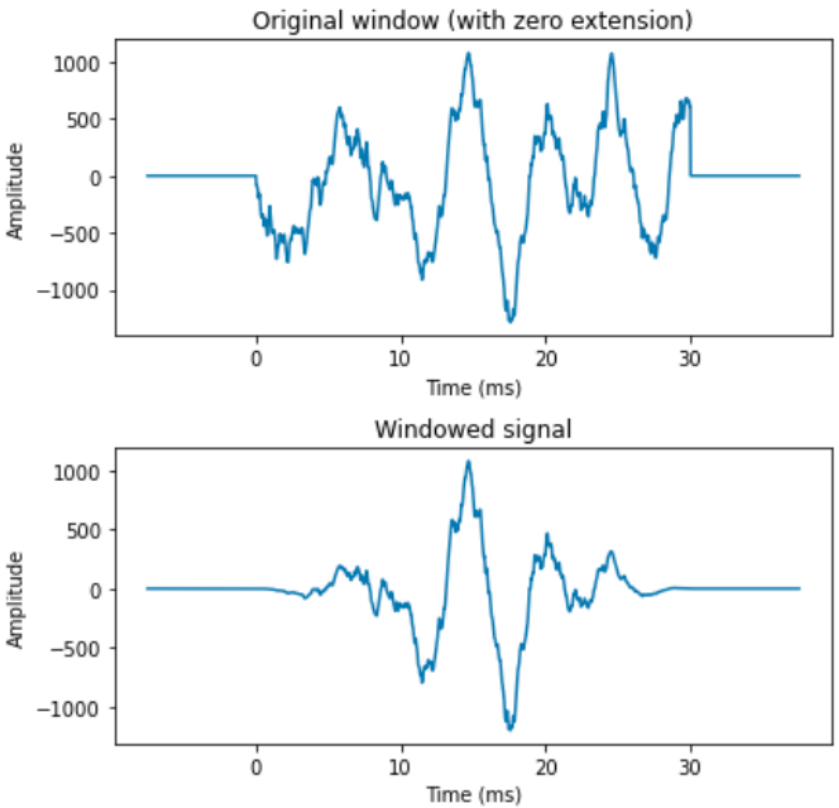
Представление звука[2]

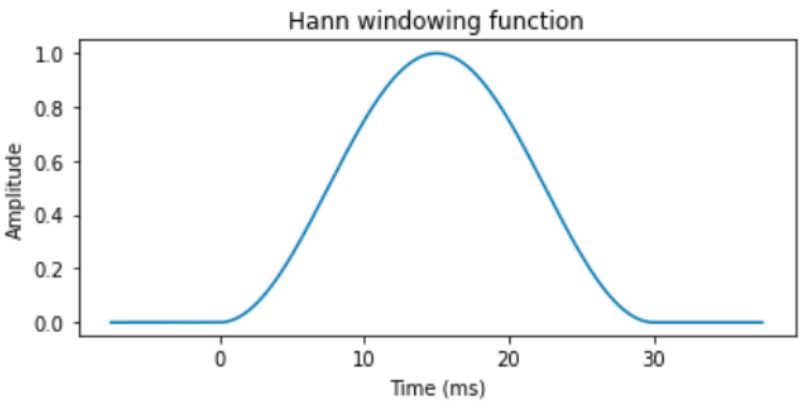


[2] Ссылка на иллюстрацию.

Обработка каждого фрагмента звука

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





Спектр

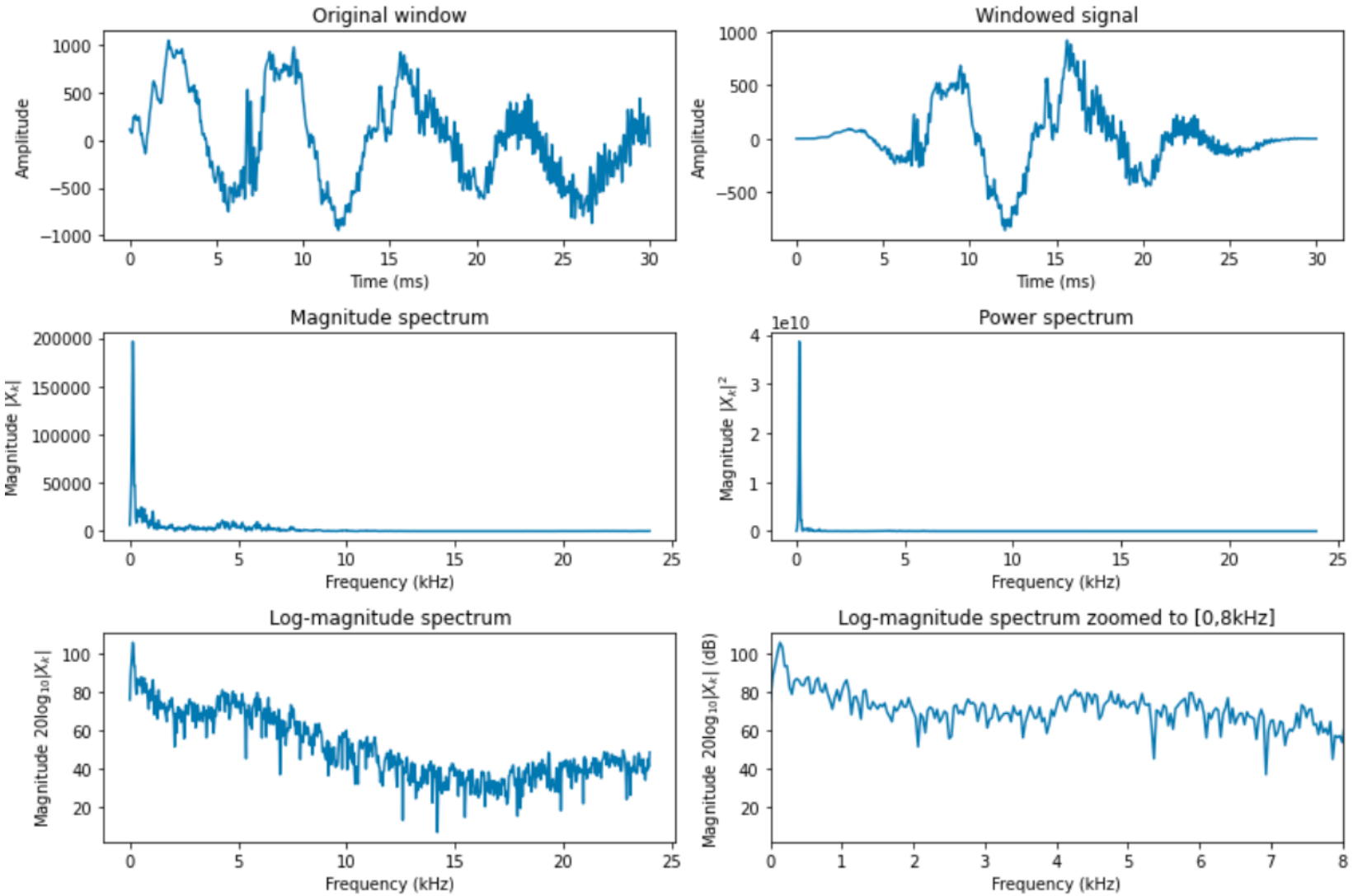
- Для извлечения характеристик каждого фрагмента используется дискретное преобразование Фурье

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

  - 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



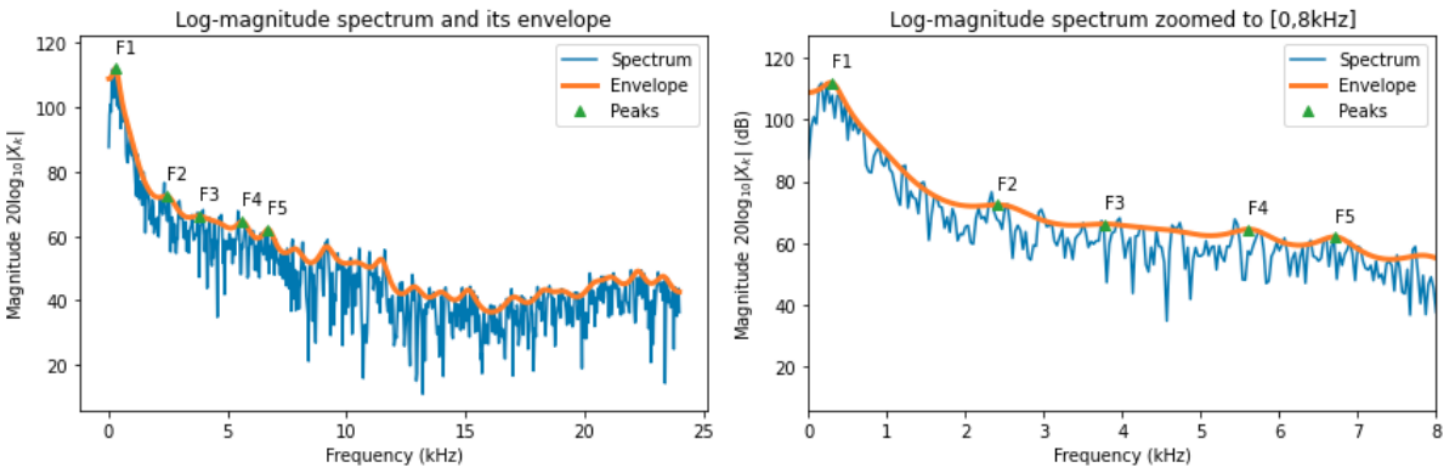
Пики на огибающей

Пики на огибающей спектра

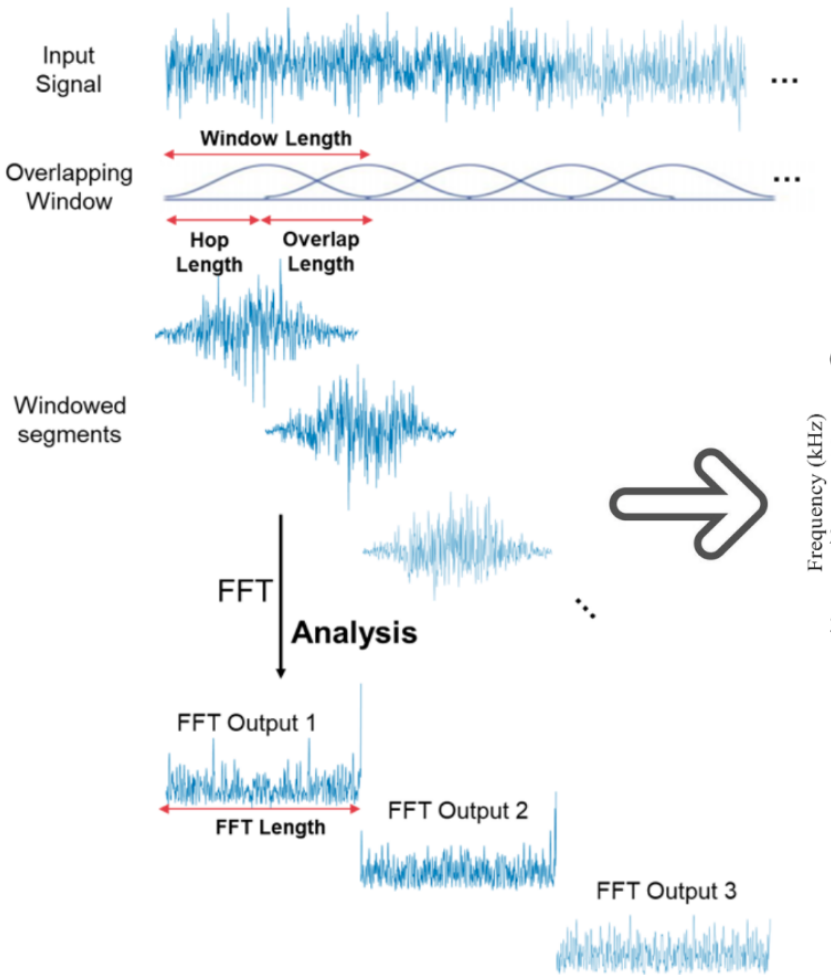
- называются формантами
- их частоты однозначно идентифицируют гласные звуки
- называются $F_1, F_2, ...$

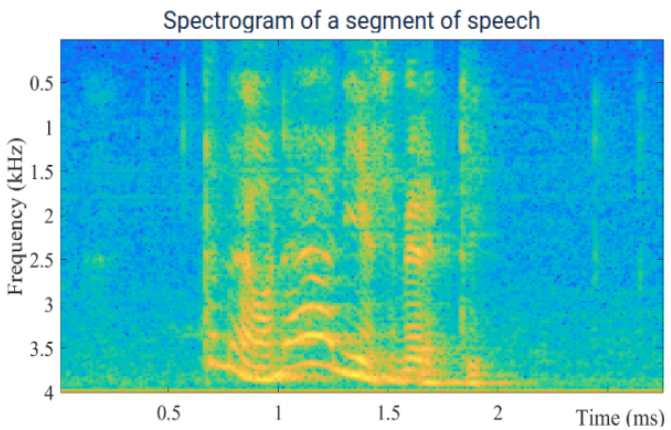
Частота колебания голосовых связок называется фундаментальной частотой F_0

- не связана с формантами и характеризует высоту голоса человека



Этапы построения спектрограммы[3]



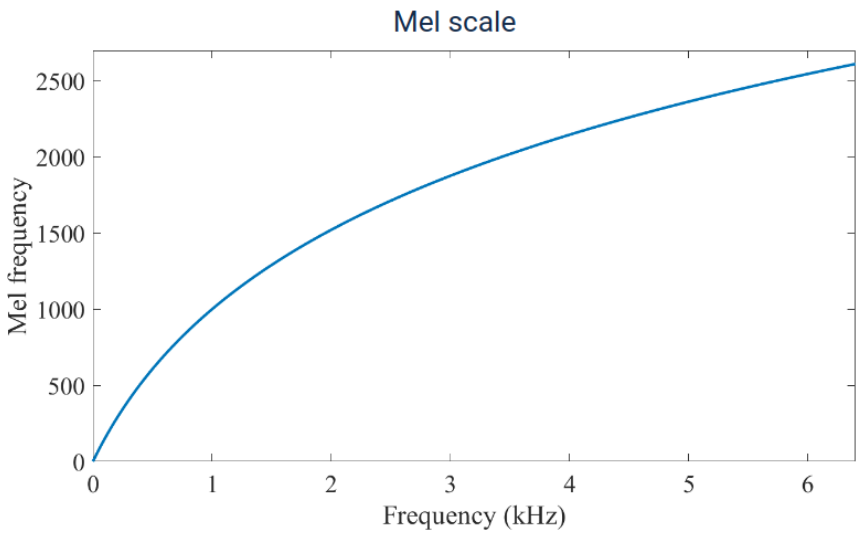


[3] Ссылка на иллюстрацию.

Мел-спектрограмма

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

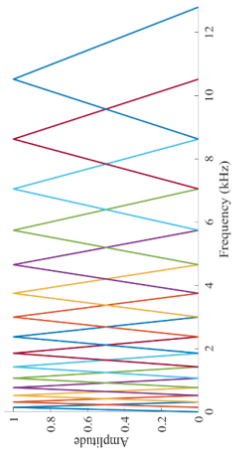
$$m = 2595 \log 10(1 + \frac{f}{700})$$

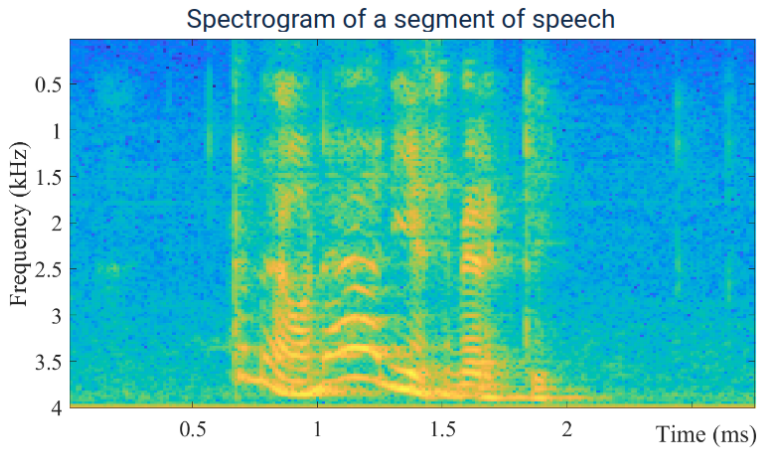


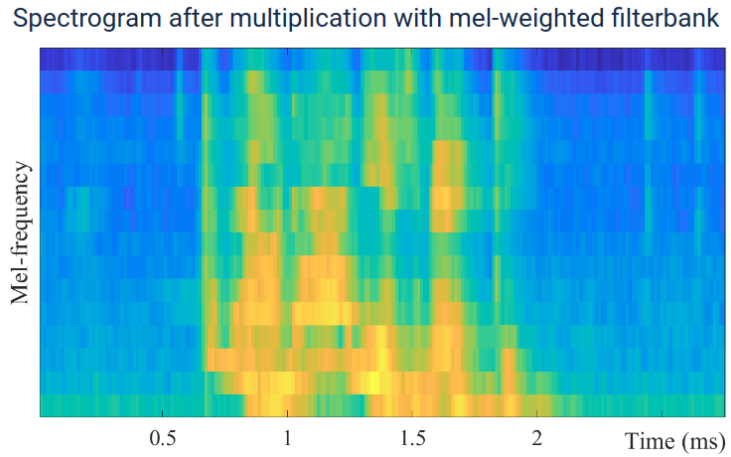
- Также #частот избыточно, можно агрегировать соседние.

Мел-спектрограмма

Усредняем соседние частоты логарифма-спектрограммы по логарифмическому закону:

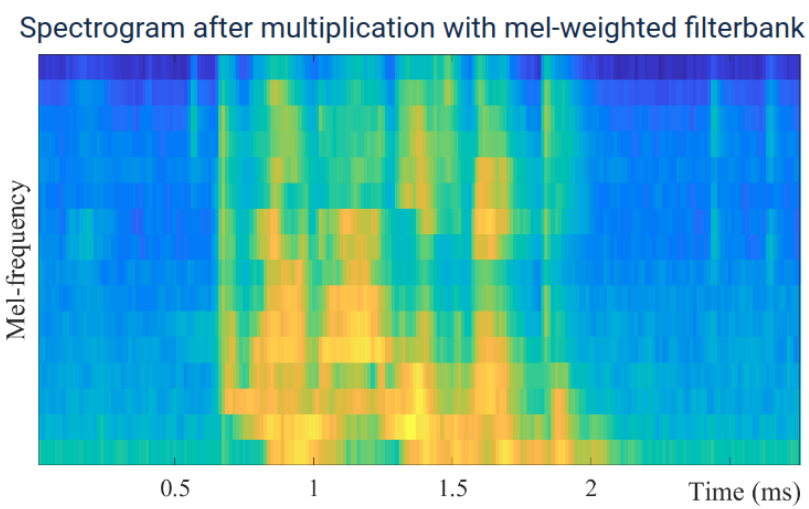


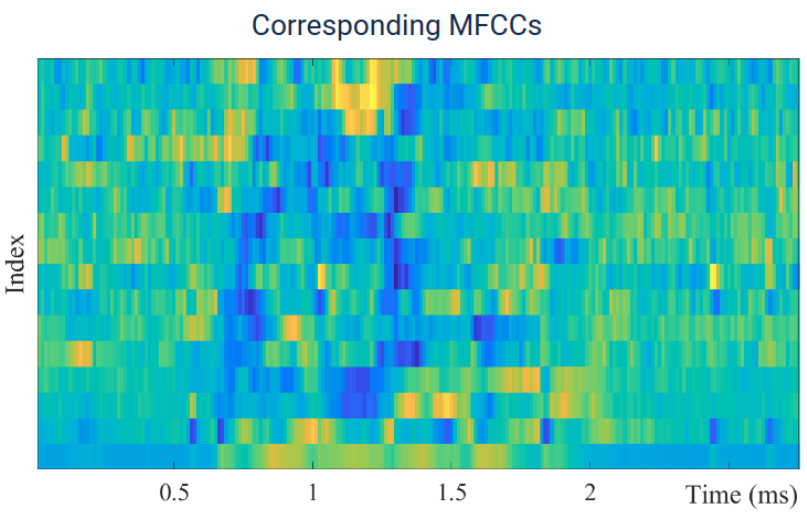




Кепстр, MFCC

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

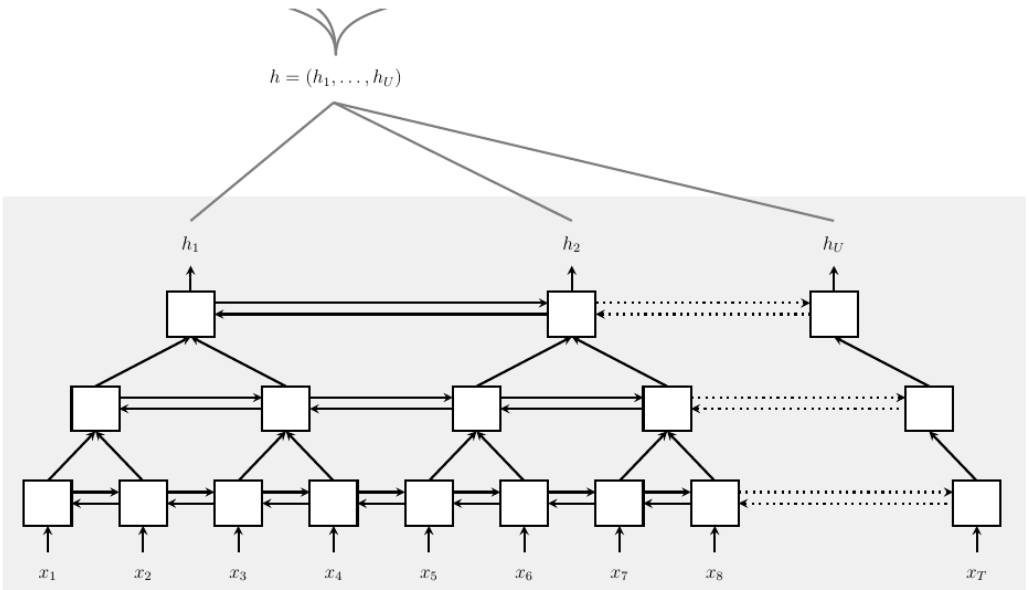




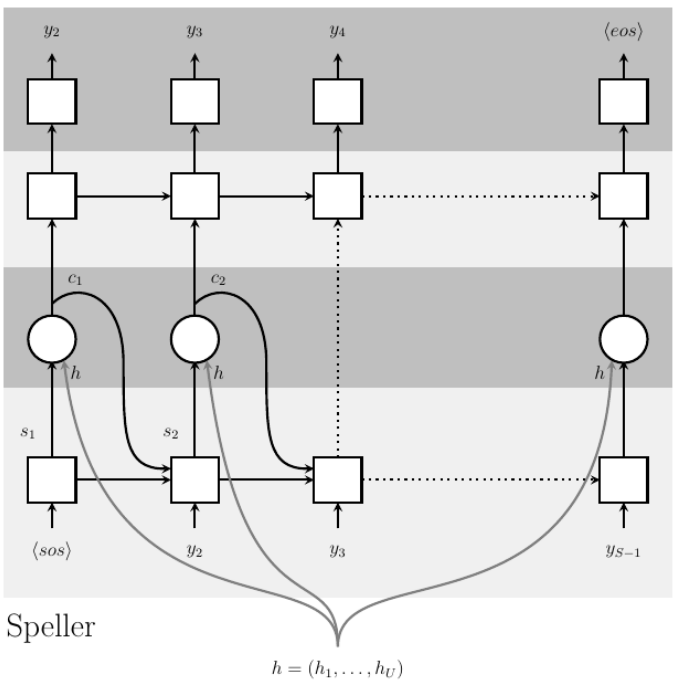
Содержание

Модель Listen-Attend-Spell[4]

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



Listener



Speller

[4] https://arxiv.org/abs/1508.01211

Модель Listen-Attend-Spell

- Декодер выдаёт распределение на символах
  - {a, b, c, $\cdots$, z, 0, $\cdots$, 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 - 3х уровневый bidirectional LSTM (BLSTM)
- Перерасчет состояния на уровне l:

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

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

Внимание в декодировщике

$$
\begin{aligned}
c_i &= \text{AttentionContext}\left(s_i, \mathbf{h}\right) \\
s_i &= \text{RNN}\left(s_{i-1}, y_{i-1}, c_{i-1}\right) \\
P\left(y_i|\mathbf{x}, y_{<i}\right) &= \text{CharacterDistribution}\left(s_i, c_i\right)
\end{aligned}
$$

- Расчёт контекста c_i:

$$
\begin{aligned}
c_i &= \sum_u \alpha_{i,u} h_u \\
\alpha_{i,u} &= \frac{\exp\left(e_{i,u}\right)}{\sum_u \exp\left(e_{i,u}\right)} \\
e_{i,u} &= \left\langle \phi\left(s_i\right), \psi\left(h_u\right) \right\rangle
\end{aligned}
$$

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

Генерация выходов

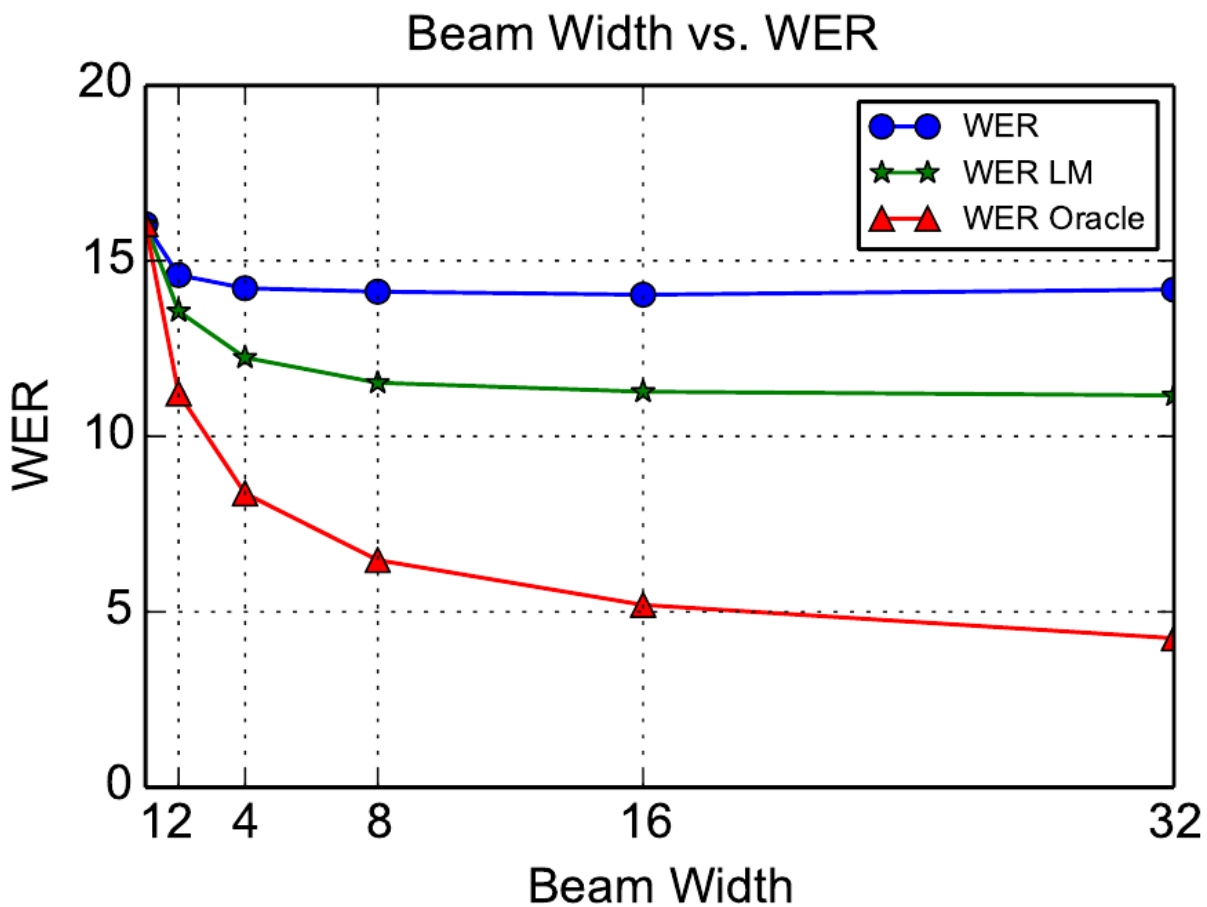
- Генерация выходной последовательности - через BeamSearch (32 лучших гипотезы)
- Считался score:

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

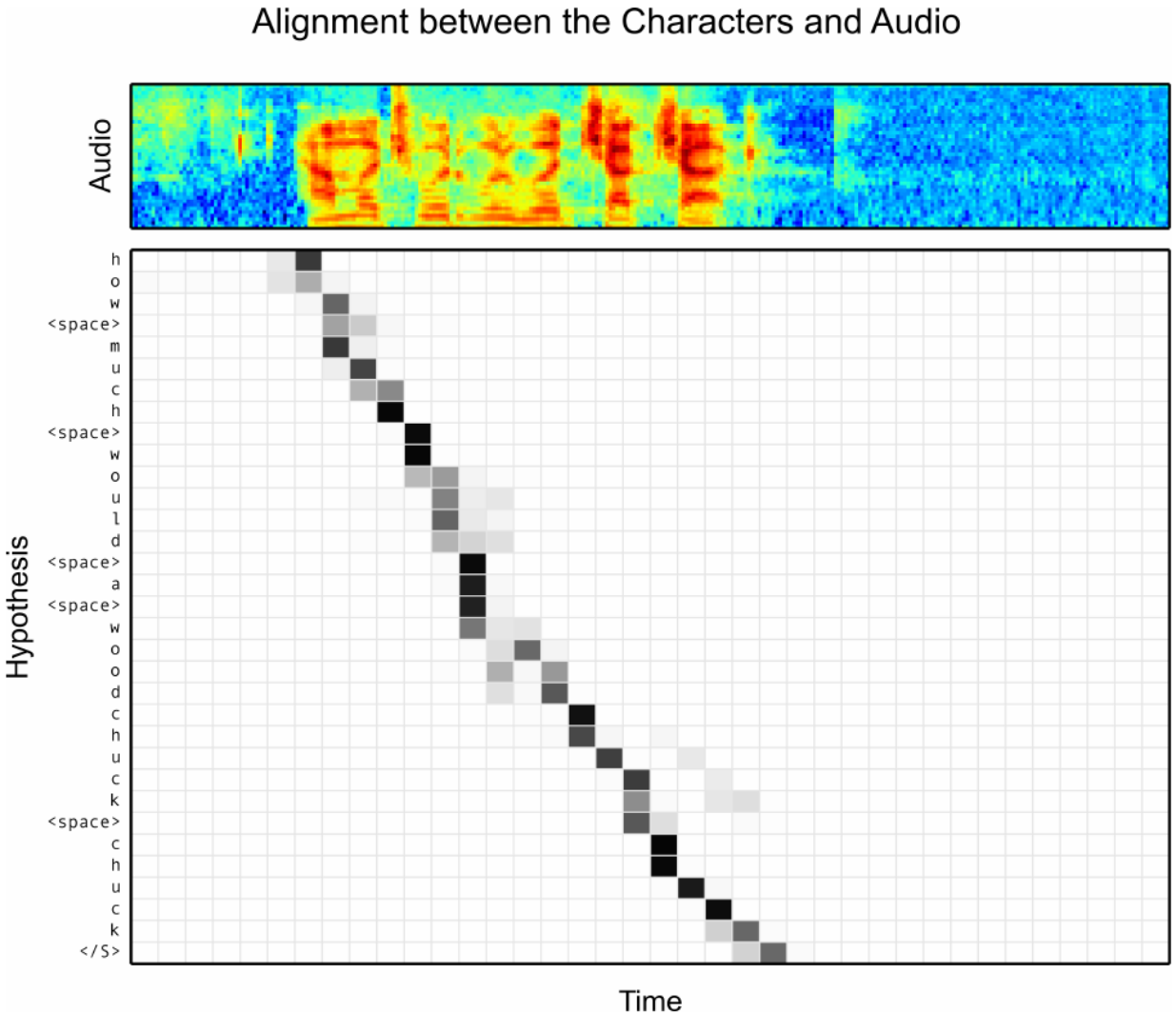
- $\log P < 0$, поэтому нормировка на #символов выхода $|y|_c$ $\uparrow$ *score*
  - чтобы поощрить модель выдавать более длинные последовательности
- $P_{LM}(\mathbf{y})$ - вероятность $\mathbf{y}$ по языковой модели.
  - много текстов для обучения
  - существенно $\uparrow$ качество
- Аугментация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

Word-error-rate от #гипотез лучевого поиска

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



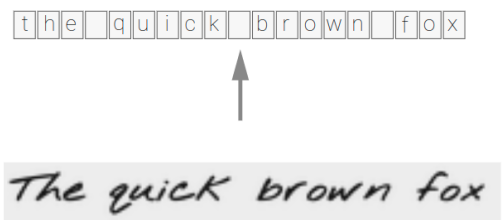
Визуализация внимания



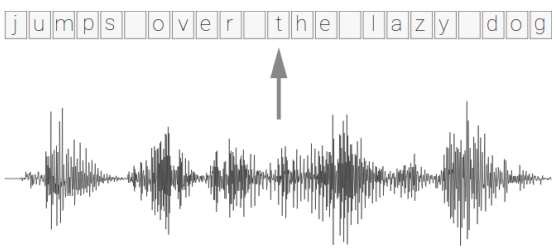
Содержание

CTC[5,6]

- Требуется построить $x_1 x_2 ... x_T \rightarrow y_1 y_s ... y_U$,
  - $T \neq U$, объекты посл-тей монотонно связаны во времени
- Примеры:
  - рукописный текст->текст
  - звук->текст
  - видео->разметка событий на фреймах



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

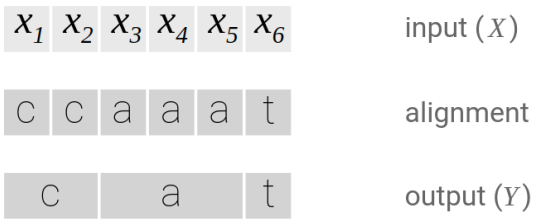


Speech recognition: The input can be a spectrogram or some other frequency based feature extractor.

[5] https://www.cs.toronto.edu/~graves/icml_2006.pdf
[6] Туториал.

Проблема выравнивания

- Для оценки модели $X \to Y$ необходимо оценивать $p(Y|X)$



- Важно построить соответствие между символами.
  - линейная связь во времени? -> нет
  - вручную -> дорого
- CTC агрегирует по всем возможным выравниваниям:

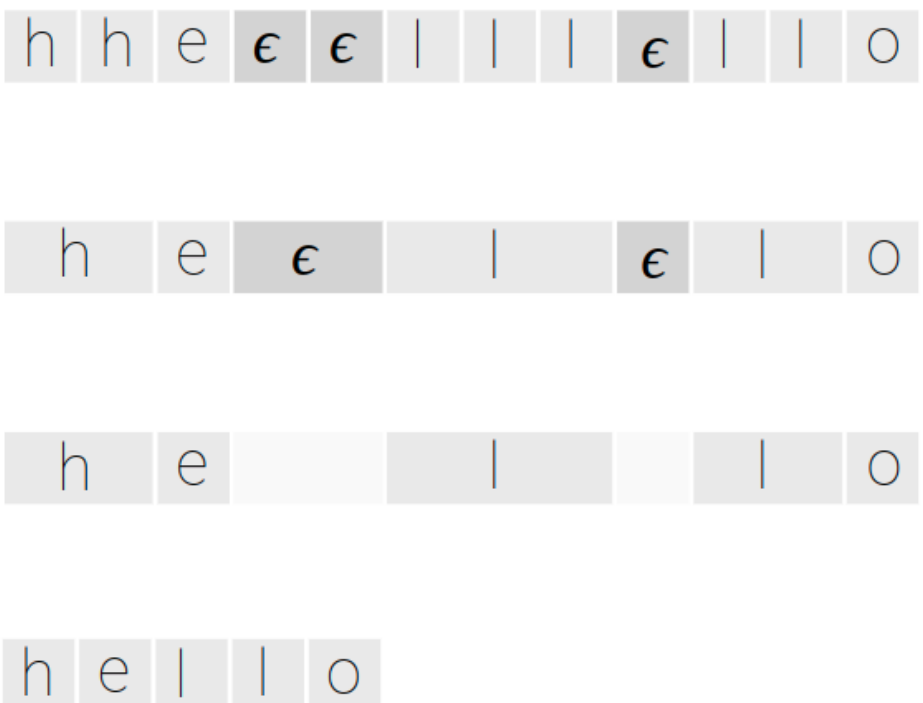
$$p(Y \mid X) = \sum_{A \in \mathcal{A}_{X,Y}} \prod_{t=1}^{T} p_t(a_t \mid X)$$

The CTC conditional **probability** | **marginalizes** over the set of valid alignments | computing the **probability** for a single alignment step-by-step.

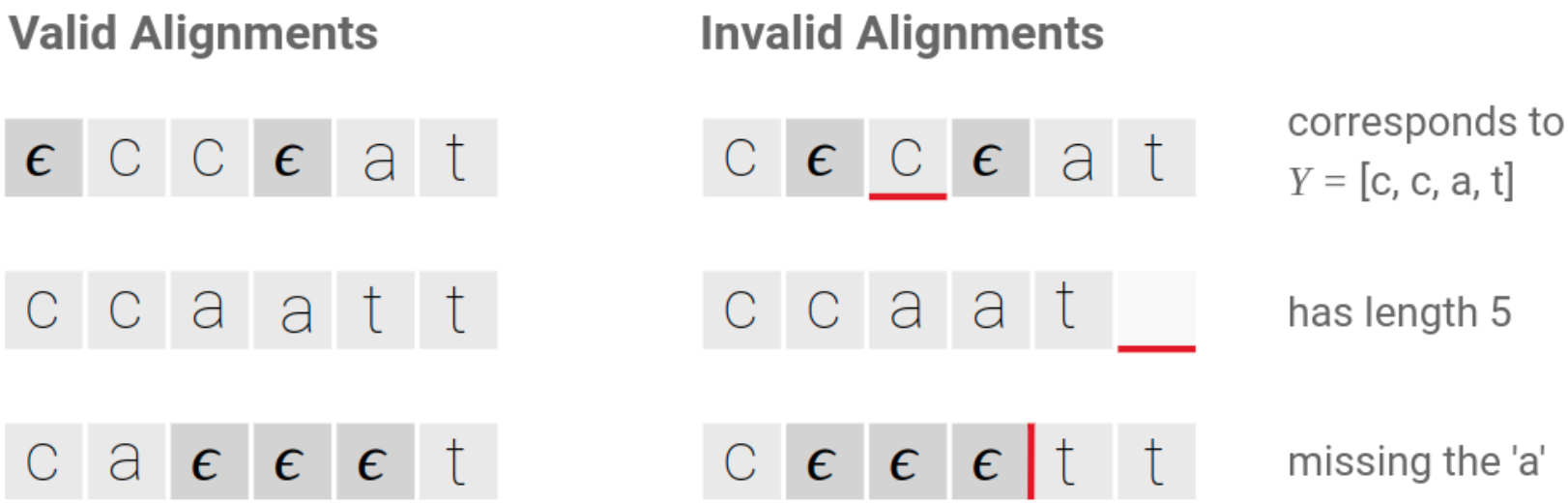
- Распределение $p(a_t|X)$ выдает модель
  - сначала CNN (conv1d или широкая свёртка), потом RNN.

Выравнивание и кодировка

- Введем пустой символ ϵ в кодировке a_t.
- Преобразование $A \to Y$:
  1. объединить повторяющиеся символы в один
  2. убрать ϵ (за счёт этого шага Y может повторять символы)

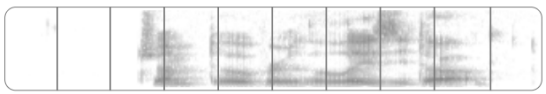


Выравнивание и кодировка

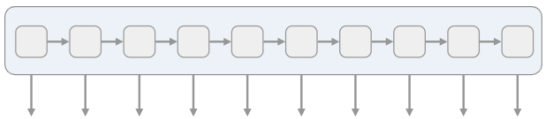


- Свойства выравнивания:
  - монотонность соответствий символов $A_t \to Y_s$
  - отображение $A_t \to Y_s$ many-to-one, $|A| \geq |Y|$

Последовательность



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



The input is fed into an RNN, for example.

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

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

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

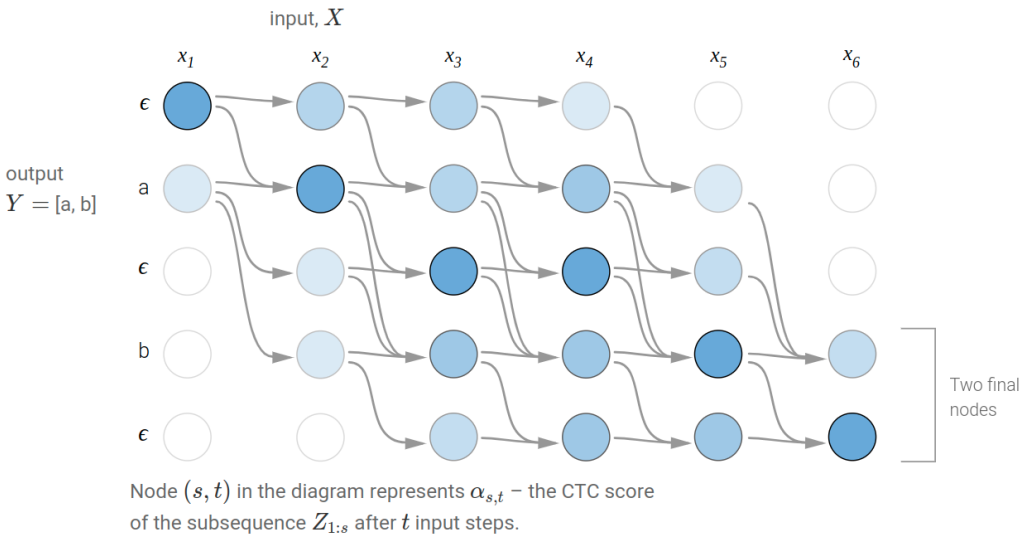
$$p(Y \mid X) = \sum_{A \in \mathcal{A}_{X,Y}} \prod_{t=1}^{T} p_t(a_t \mid X)$$

The CTC conditional **probability**

marginalizes over the set of valid alignments

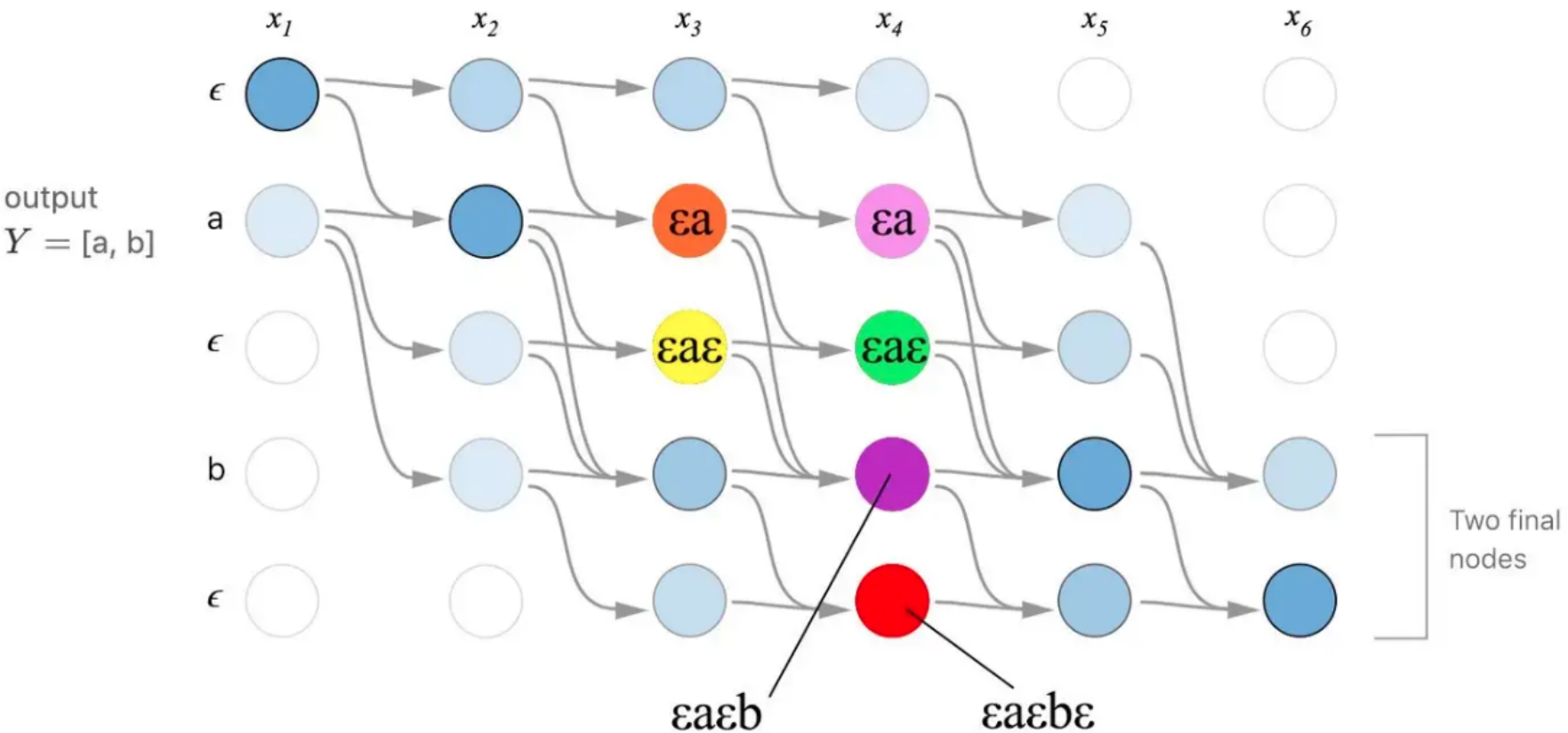
computing the **probability** for a single alignment step-by-step.

Агрегация по выравниваниям



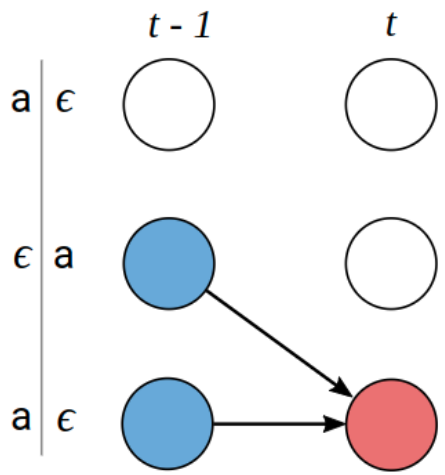
- По строкам $Z = [\epsilon, y_1, \epsilon, y_2, \epsilon, ...\epsilon, y_N, \epsilon]$
  - $Y =' \textbf{hello}' \rightarrow Z =' \epsilon\textbf{h}\epsilon\textbf{e}\epsilon\textbf{l}\epsilon\textbf{l}\epsilon\textbf{o}\epsilon'$
- По столбцам - входы (фреймы аудио)
- $\alpha_{s,t} = p\left(Z_{1:s}|X_{1:t}\right)$

Агрегация по выравниваниям



Случай 1

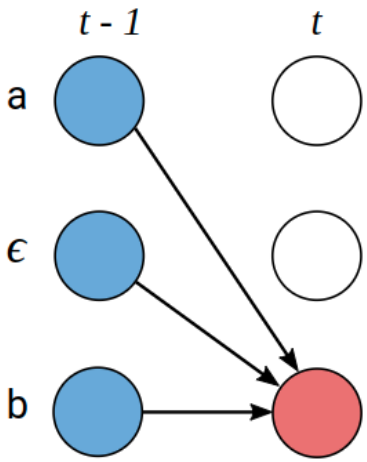
In this case, we can't jump over z_{s-1}, the previous token in Z. The first reason is that the previous token can be an element of Y, and we can't skip elements of Y. Since every element of Y in Z is followed by an ϵ, we can identify this when $z_s = \epsilon$. The second reason is that we must have an ϵ between repeat characters in Y. We can identify this when $z_s = z_{s-2}$.



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

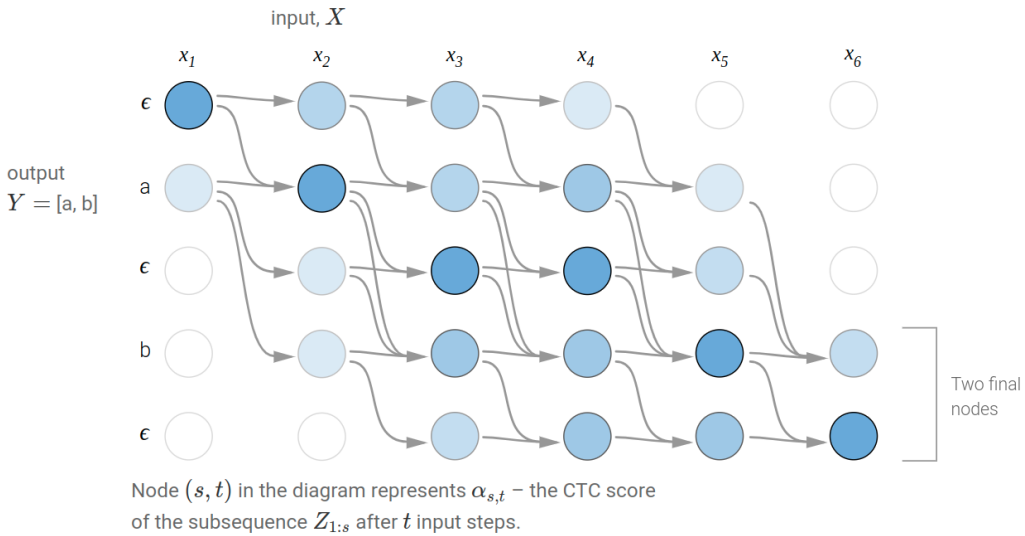
Случай 2

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.



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

Пример выравнивания



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

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

Построение прогноза (наивный подход)

Построение прогноза

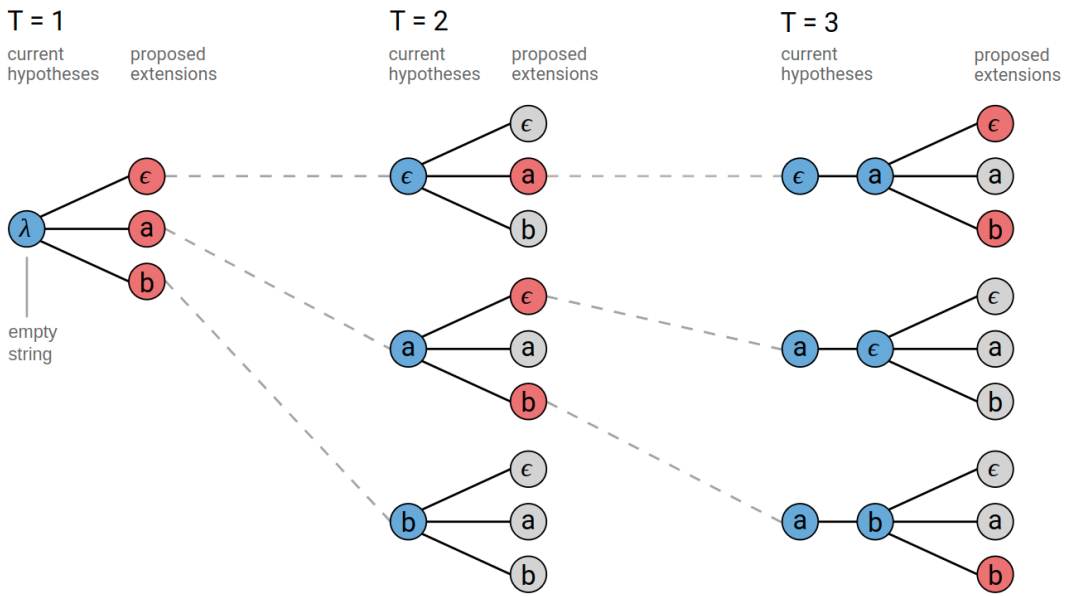
$$\widehat{Y} = \arg\max_{Y} p\left(Y|X\right)$$

Простой прогноз:

$$\widehat{A} = \arg\max_{A} \prod_{t=1}^{T} p\left(a_t|X\right)$$

$\widehat{A} \to \widehat{Y}$ через CTC преобразование

Построение прогноза (наивный подход)



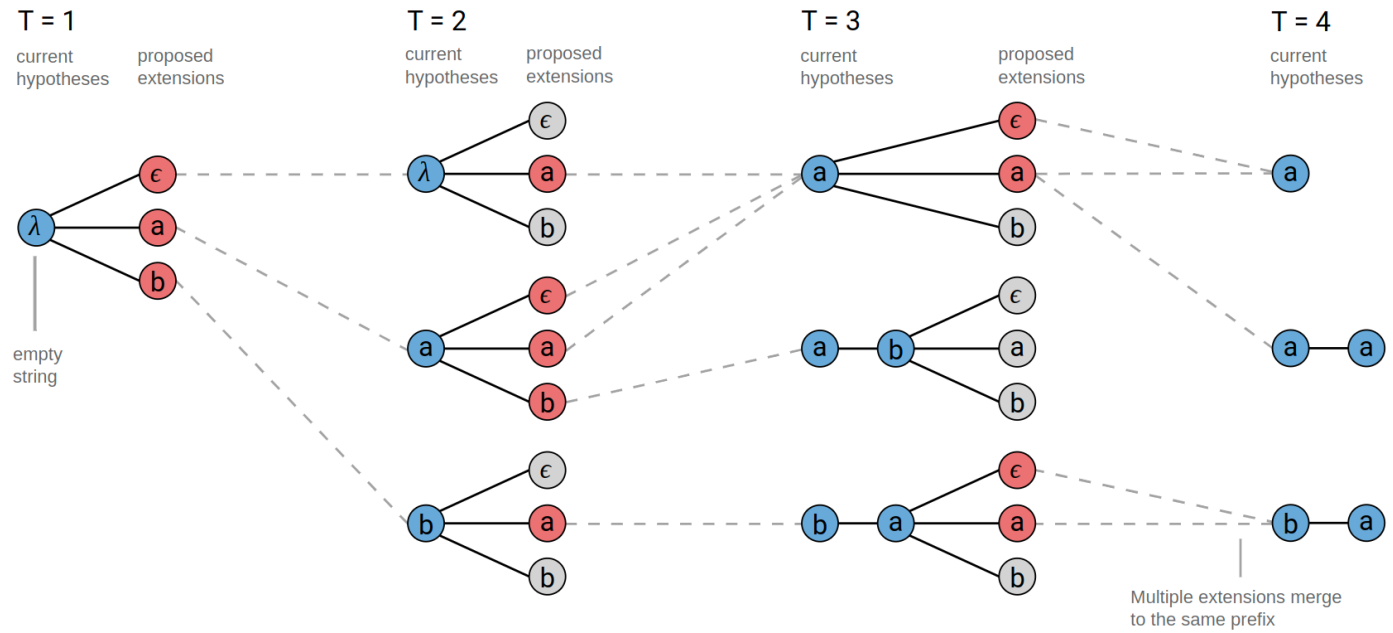
A standard beam search algorithm with an alphabet of $\{\epsilon, a, b\}$ and a beam size of three.

Недостаток простого подхода: разные выравнивания могут соответствовать одному выходу

- например $p(bbb) > p(aa\epsilon)$ и $p(bbb) > p(aaa)$, но $p(aa\epsilon) + p(aaa) > p(bbb)$ и нужно выдавать $\widehat{Y} = a$.

Построение прогноза с учетом выходного Y

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



The CTC beam search algorithm with an output alphabet $\{\epsilon, a, b\}$ and a beam size of three.

- Для этого агрегируем

$$\epsilon a + a\epsilon + aa \to a, \epsilon aa + \epsilon a\epsilon \to a, baa + ba\epsilon \to ba, \ldots$$

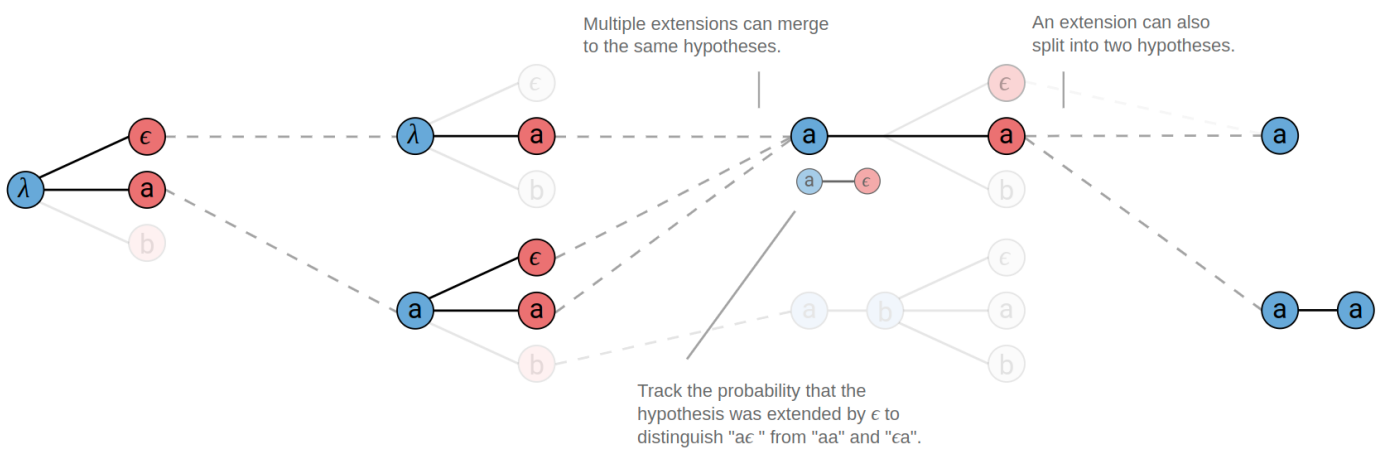
Построение прогноза с учетом выходного Y

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

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

- При склейке же с b результат одинаковый:

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



Повышение качества прогнозов

Улучшенный рейтинг для ↑ качества выходов:

$$p(Y|X) \cdot p(Y)^{\alpha} \cdot |Y|^{\beta}$$

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

Проверка корректной реализации CTC:

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

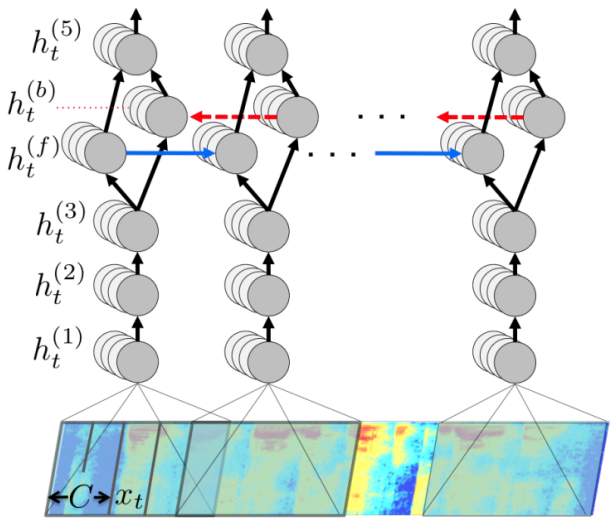
Должно получиться:

$$p_{CTC,\beta}(Y|X) \leq p_{true}(Y|X)$$

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

DeepSpeech[7]

- Модель DeepSpeech реализует CTC.
- X_t - окно спектрограммы (+- 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:



[7] 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:

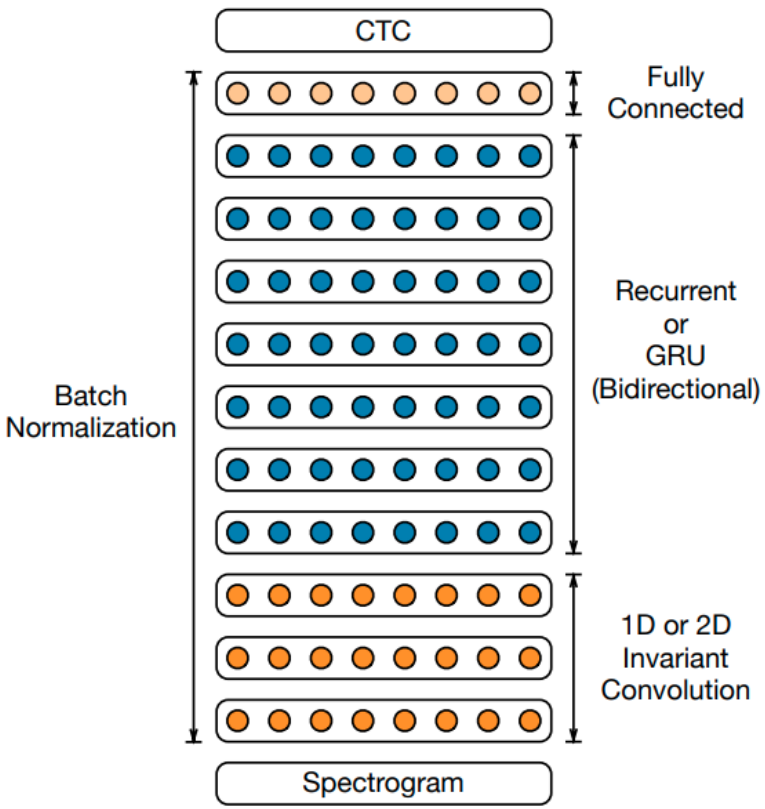
$$\frac{1}{3}\left(F(x_{-5ms}) + F(x) + F(x_{+5ms})\right)$$

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

$$\mathrm{ReL}U_{clip}(x) = \min\left\{\max\left\{x, 0\right\}, 20\right\}$$

DeepSpeech 2[8]

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



[8] https://arxiv.org/pdf/1512.02595.pdf

DeepSpeech 2 - детали архитектуры

- свёрточные слои (2D-conv по времени и частоте лучше себя показала)
- 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 CTC: DeepSpeech, DeepSpeech 2.
- качество $\uparrow$, если
  - используем языковую модель
  - поощряем длительность y
  - используем аугментацию
    - добавляется эхо
    - учимся на x чуть смещенных по t
    - добавляем посторонний шум