CS554 Capstone I - Fall 2021

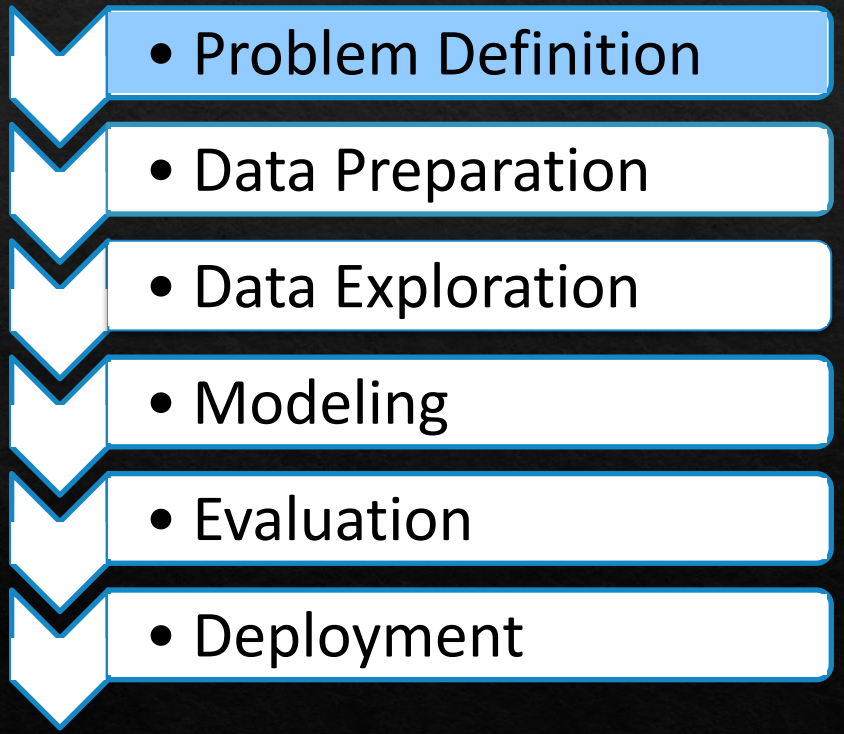
Podcast Speech to Text transcription

Supervised by -

Prof. Saed Sayad

Abhishek Bhatt (ab2083)

Data Mining - 6 steps



Introduction

- As per the internet, there are > 2M podcasts and > 48M episodes as of 2021
  - impossible to listen to them all for a single user

- As opposed to mainstream media, contain invaluable information
  - expert analyses and diverse viewpoints, ranging from science, history, society and beyond

Problem Definition

- Create an easy to explore knowledge repository from full-length audio podcasts
  - Split podcasts into smaller speaker segments
  - **Build a model to transcribe audio segments to text**
  - Store the text segments in a searchable format
  - Summarization, entity extraction over text

Criteria of success

Word Error Rate (WER)

- how many “errors” in transcription text, compared to a human transcription
- De facto standard for Automatic Speech Recognition
- **The lower the WER the better**

$$WER = \frac{S + D + I}{N}$$

Word Error Rate Algorithm

What we are doing here is combining the number of **Substitutions (S)**, **Deletions (D)**, and **Insertions (N)**, divided by the **Number of Words (N)**.

Criteria of success

Word Error Rate (WER)

$$WER = \frac{S + D + I}{N}$$

- “Hello there”
  hello_
  1 deletion => WER=50%

- “Hi my name is Bob and I like cheese. Cheese is very good.”
  hi my **frame** is **knob** and i **bike leafs** cheese is **berry wood**
  6 substitutions => WER = 46%

  hi my name is bob and i **bike** cheese cheese is_good **i know**
  1 substitution, 1 deletion, 2 insertions => WER = 30%

Criteria of success

Word Error Rate (WER)

```
Human transcription:
"I live in New York"


Automatic transcription:
"i live in new york"
WER: 60%
```

- Lowercase all text
- Remove all punctuation
- Change all numbers to their written form: "7" -> "seven"

```
Spoken:
"I like to bike around"

Model 1 prediction:
"I liked to bike around"
WER: 20%

Model 2 prediction:
"I like to bike pound"
WER: 20%
```

Moving from word to character level

- Character Error Rate (CER)

Criteria of success

Levenshtein distance

- Difference between two words based on how many characters need to be changed to get there
- More in-depth look at what changes are being made, rather than just how many changes happen
- **The smaller the Levenshtein distance, the smaller is the difference between the strings**

For example, the Levenshtein distance between **bird** and **bard** is one. The difference between **kitten** to **sitting** would be three because you'd have to make three substitutions to get there.

1. **k**itten → **s**itten (substitution of "s" for "k")
2. sitt**e**n → sitt**i**n (substitution of "i" for "e")
3. sittin → sittin**g** (insertion of "g" at the end).

Criteria of success

Character Error Rate (CER = LD / N)

$$CER = \frac{S + D + I}{N}$$

"i like to bike around"

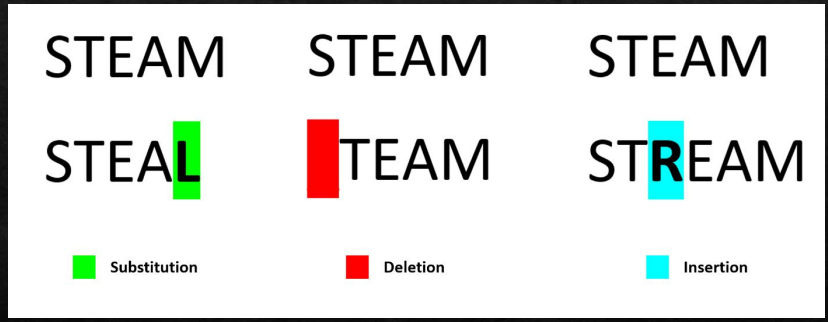
- i liked to bike around

  1 insertion => CER = 1/21 = 4.8%

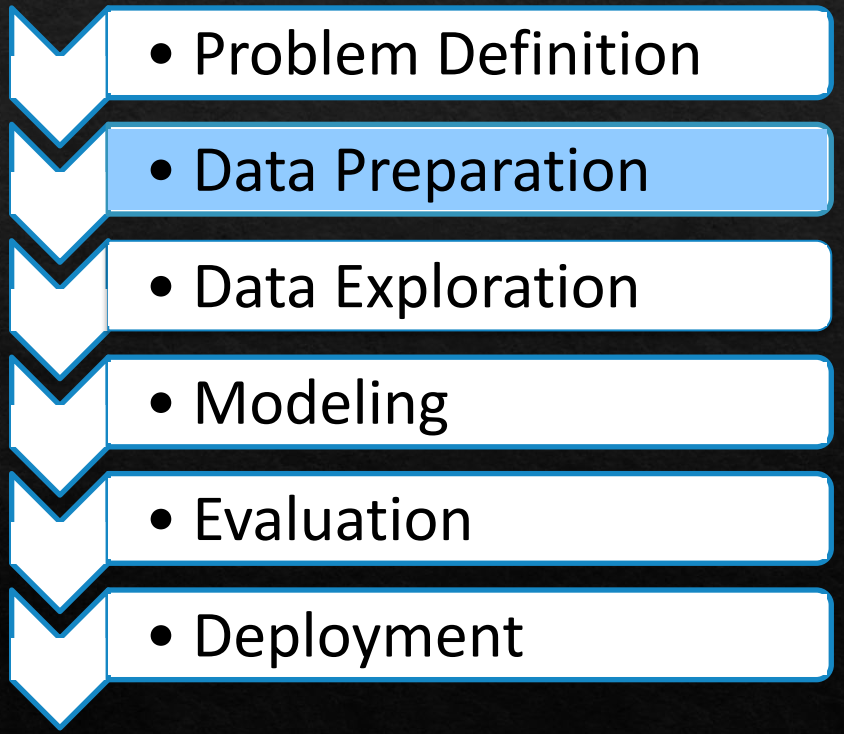
- i like to bike _pound

  1 deletion, 1 substitution => CER = 2/21 = 9.5%

- WER in both cases = ⅕ = 20%



Data Mining - 6 steps



Dataset Overview

[Lex Fridman Podcast playlist on Youtube]

➢ 200+ episodes, each ranging from 2 to 4 hours.

Raw data

- **Audio files**

Metadata

- **Text**

Transcripts as ground truth for modeling

- **JSON text (Youtube English closed captions)**

Raw Data Collection

- *pytube* - fetch mp4 file, metadata and closed captions from Youtube
- *ffmpeg* - convert downloaded mp4 to wav file
  - Format : RIFF (little-endian) data, WAVE audio, Microsoft PCM, 16 bit, mono 16000 Hz

```
(base) ab2083@ilab2:~/Capstone/podcast_LDTe8uFqbws$ ls
'Frank Wilczek_ Physics of Quarks_ Dark Matter_ Complexity_ Life _ Aliens _ Lex Fridman Podcast _187-cc.txt'
'Frank Wilczek_ Physics of Quarks_ Dark Matter_ Complexity_ Life _ Aliens _ Lex Fridman Podcast _187.txt'
'Frank Wilczek_ Physics of Quarks_ Dark Matter_ Complexity_ Life _ Aliens _ Lex Fridman Podcast _187.wav'
(base) ab2083@ilab2:~/Capstone/podcast_LDTe8uFqbws$
```

x 212 episodes

Data Segmentation

TITLE.WAV

Full length episode

TITLE-CC.TXT

[{"text": "helped educate and inspire millions of", "start": 27.63, "duration": 6.07} , ... {}]

x 212 episodes

pydub

(audio, start_time, end_time)

X_1 -> TITLE#start_time1#end_time1.WAV

Y_1 -> TITLE#start_time1#end_time1.TXT

X_K -> TITLE#start_timeK#end_timeK.WAV

Y_K -> TITLE#start_timeK#end_timeK.TXT

X -> AUDIO

Y -> TEXT

Data Snippet

- **534, 974 (X : audio, Y : text) pairs** extracted from 212 episodes
  - Ignored segments where Y_i has less than 5 words
  - No punctuation, all lowercase alphabets in all Y_i

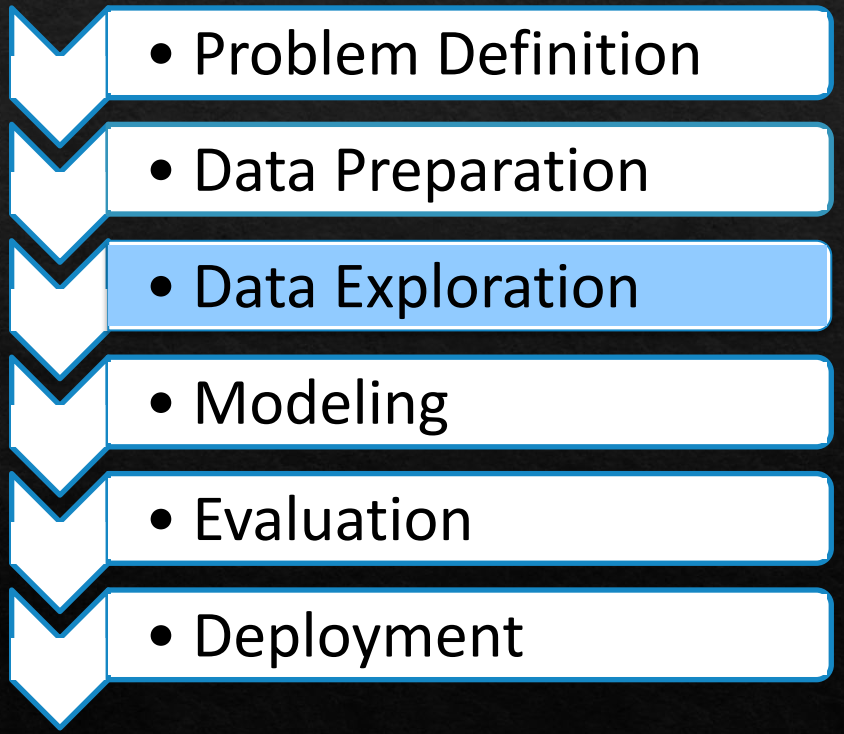
Host - Lex (Male)	Guest - Male	Guest - Female
Text - history of the 20th century in russia	Text - predator like a large cat or something	Text - like 100 electron volts in terms of the
		

Data Split

TRAINING SET	VALIDATION SET	TEST SET	TOTAL
80 %	10 %	10 %	100%
428, 105 (X, Y) pairs	53, 401 (X, Y) pairs	53, 468 (X, Y) pairs	534, 974 (X, Y) pairs

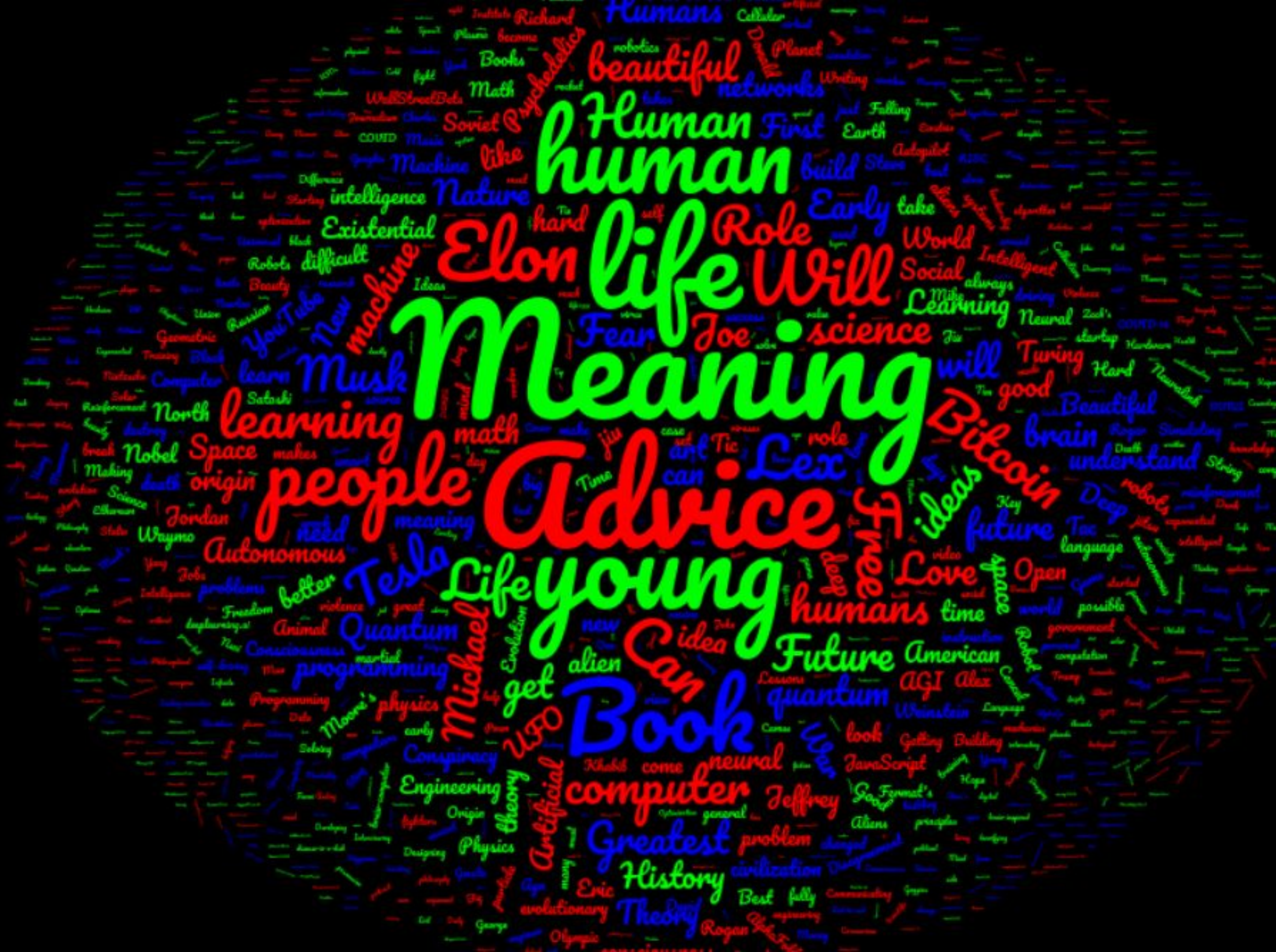
x 212 episodes

Data Mining - 6 steps



Metadata -

Key Ideas



Metadata - *Episodes*

	titles	ratings	lengths	views
0	Chris Duffin_ The Mad Scientist of Strength _ Lex Fridman Podcast _207	4.895561	9777	173399
1	Jason Calacanis_ Startups_ Angel Investing_ Capitalism_ and Friendship _ Lex Fridman Podcast _161	4.838706	7870	246100
2	David Fravor_ UFOs_ Aliens_ Fighter Jets_ and Aerospace Engineering _ Lex Fridman Podcast _122	4.767675	14184	4881542
3	Sebastian Thrun_ Flying Cars_ Autonomous Vehicles_ and Education _ Lex Fridman Podcast _59	4.919451	4714	60562
4	Cristos Goodrow_ YouTube Algorithm _ Lex Fridman Podcast _68	4.832593	5452	43164
...	...	...	...	...
207	Leslie Kaelbling_ Reinforcement Learning_ Planning_ and Robotics _ Lex Fridman Podcast _15	4.946349	3683	21993
208	Gilbert Strang_ Linear Algebra_ Teaching_ and MIT OpenCourseWare _ Lex Fridman Podcast _52	4.961246	2993	139446
209	Garry Kasparov_ Chess_ Deep Blue_ AI_ and Putin _ Lex Fridman Podcast _46	4.899026	3324	325772
210	Eugenia Kuyda_ Friendship with an AI Companion _ Lex Fridman Podcast _121	4.862124	10866	188990
211	Yann LeCun_ Deep Learning_ ConvNets_ and Self_Supervised Learning _ Lex Fridman Podcast _36	4.933997	4558	117368

212 rows × 4 columns

	ratings	lengths	views
count	212.000000	212.000000	2.120000e+02
mean	4.884365	7038.349057	3.563384e+05
std	0.061410	2883.282284	5.610591e+05
min	4.524468	1965.000000	1.784700e+04
25%	4.862245	4806.250000	7.606525e+04
50%	4.897732	6622.500000	1.515580e+05
75%	4.922474	9304.000000	3.528318e+05
max	4.977918	15319.000000	4.881542e+06

Metadata - *Guests*

```
pd.set_option('display.max_colwidth', None)
df_guests
```

	name	bio
0	Chris Duffin	Chris Duffin is a strength athlete, coach, and engineer, setting multiple strength world records including squatting and deadlifting 1000 lbs for multiple reps
1	Jason Calacanis	Jason Calacanis is an angel investor, entrepreneur, and co-host of All-In Podcast and This Week in Startups
2	David Fravor	David Fravor is a navy pilot of 18 years and a primary witness in one of the most credible UFO sightings in history
3	Sebastian Thrun	NaN
4	Cristos Goodrow	Cristos Goodrow is VP of Engineering at Google and head of Search and Discovery at YouTube (aka YouTube Algorithm)
...	...	...
207	Leslie Kaelbling	NaN
208	Gilbert Strang	NaN
209	Garry Kasparov	NaN
210	Eugenia Kuyda	Eugenia Kuyda co-founder of Replika, an AI companion
211	Yann LeCun	NaN

212 rows × 2 columns

```
df_guests.isna().sum()
```

```
name     0
bio     61
dtype: int64
```

Audio Exploration / Feature Extraction

- Conventional modeling for audio used hand-crafted features (example on the right)
- MFCC / Spectrogram - features capturing the "content" of the audio rather than nature of the speaker; mimic human speech perception
  - Useful for modern deep learning approaches

filename	blues.00000.wav
length	661794
chroma_stft_mean	0.350088
chroma_stft_var	0.088757
rms_mean	0.130228
rms_var	0.002827
spectral_centroid_mean	1784.16585
spectral_centroid_var	129774.0645
spectral_bandwidth_mean	2002.44906
spectral_bandwidth_var	85882.76132
rolloff_mean	3805.839606
rolloff_var	9.02E+05
zero_crossing_rate_mean	0.083045
zero_crossing_rate_var	0.000767
harmony_mean	-4.53E-05
harmony_var	0.008172
perceptr_mean	0.000008
perceptr_var	0.005698
tempo	123.046875
mfcc1_mean	-113.570648

Amplitude Envelope

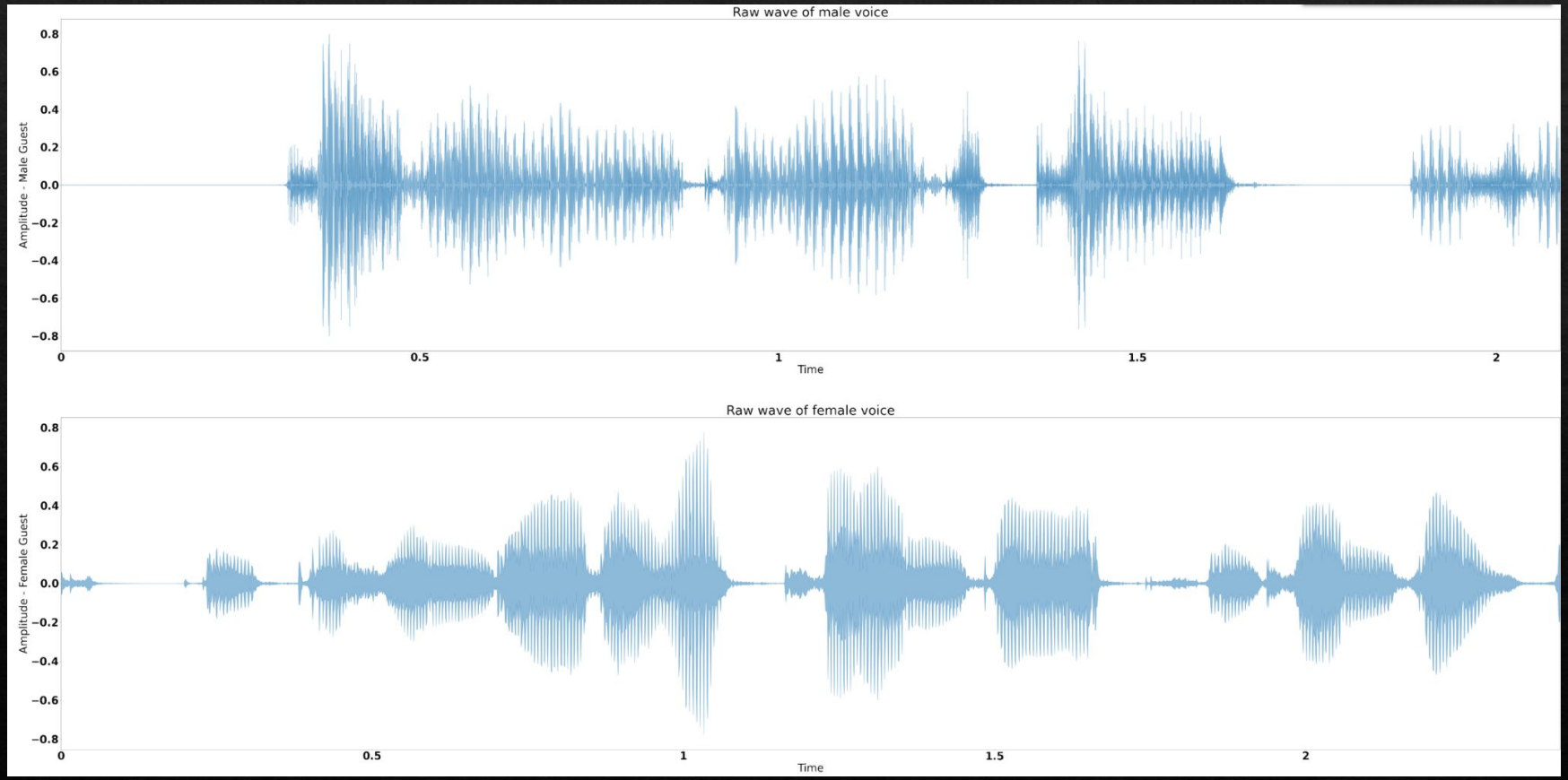
Amplitude Envelope

- Outlines the extremes of the signal amplitude
- *Amplitude* - **loudness** or max displacement of vibrating medium particles of the medium from their mean position

```
y_m, sr_m = librosa.load(AUDIO_M)
y_f, sr_f = librosa.load(AUDIO_F)
```

```
plt.figure(figsize=(100, 50))

plt.subplot(2, 1, 1)
plt.title('Raw wave of male voice')
plt.ylabel('Amplitude - Male Guest')
librosa.display.waveplot(y_m, sr=sr_m, alpha=0.5)
```

Amplitude Envelope



Spectrogram

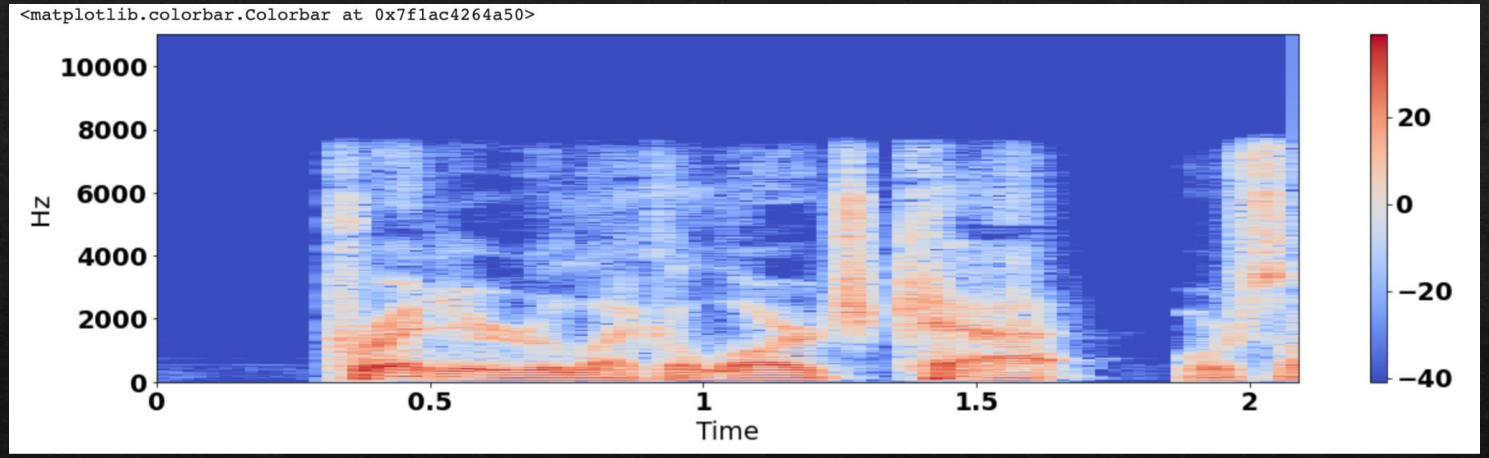
Spectrogram

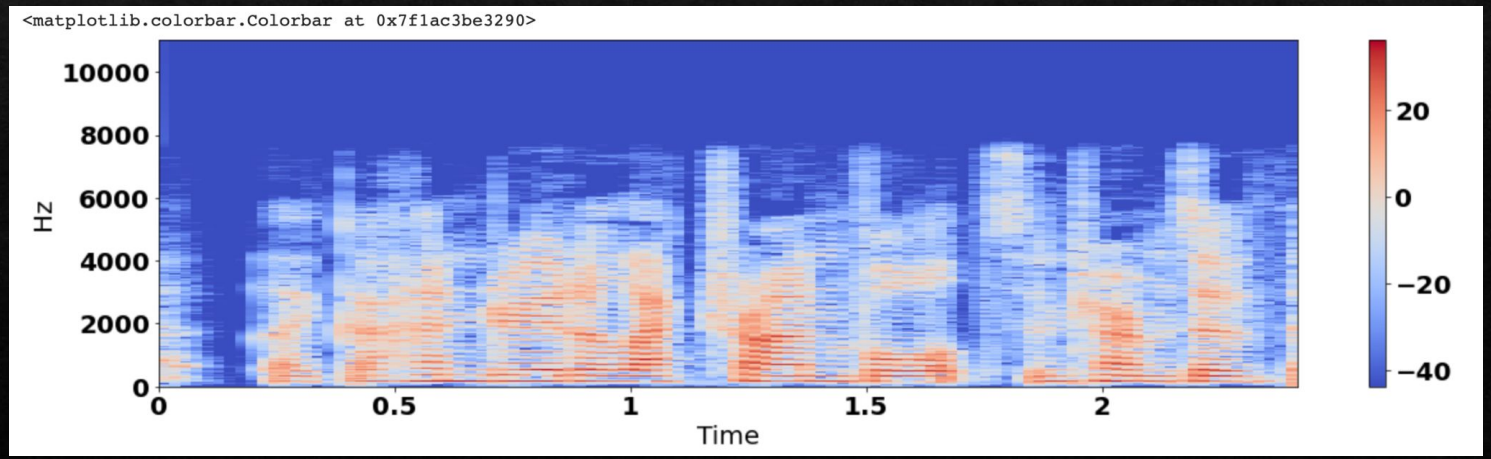
- Spectrum of **frequencies** of a signal as it varies with time

```
#spectogram - male voice (hz)

X_m = librosa.stft(y_m)
Xdb_m = librosa.amplitude_to_db(abs(X_m))
plt.figure(figsize=(20,5))
librosa.display.specshow(Xdb_m, sr=sr_m, x_axis='time', y_axis='hz')
plt.colorbar()
```

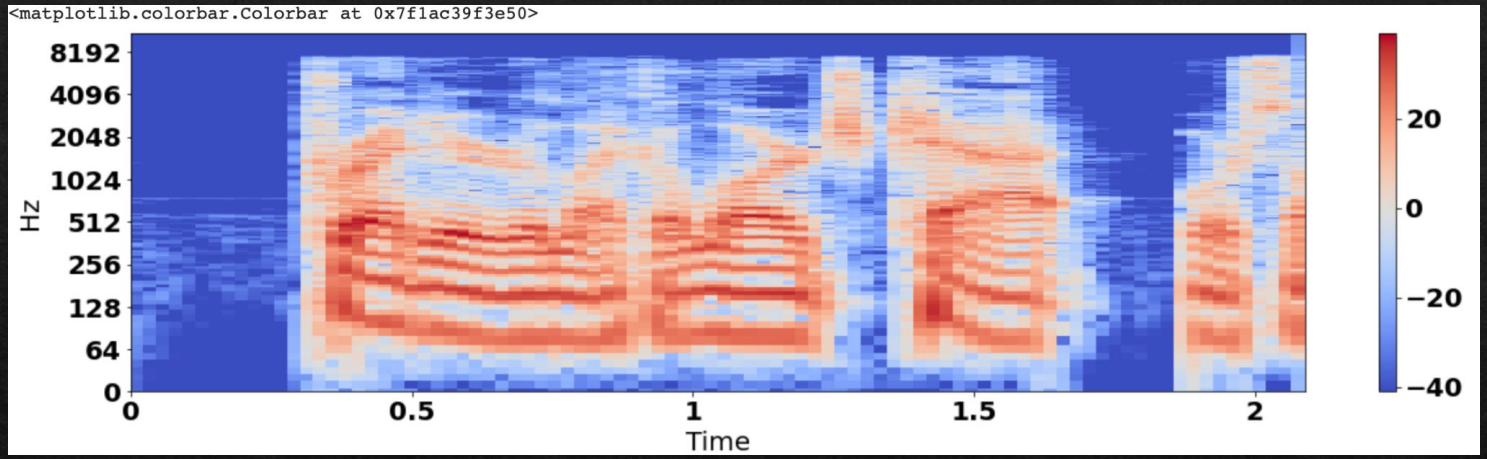
```
#spectogram - male voice (log)

X_m = librosa.stft(y_m)
Xdb_m = librosa.amplitude_to_db(abs(X_m))
plt.figure(figsize=(20,5))
librosa.display.specshow(Xdb_m, sr=sr_m, x_axis='time', y_axis='log')
plt.colorbar()
```

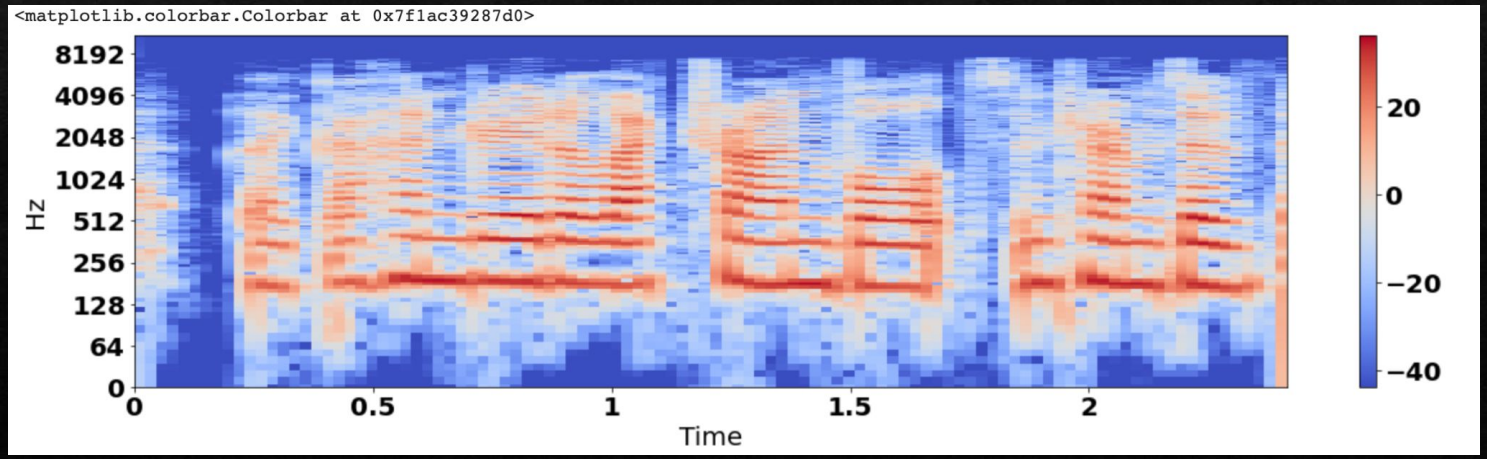
Spectrogram (Hz)





Spectrogram (Log)





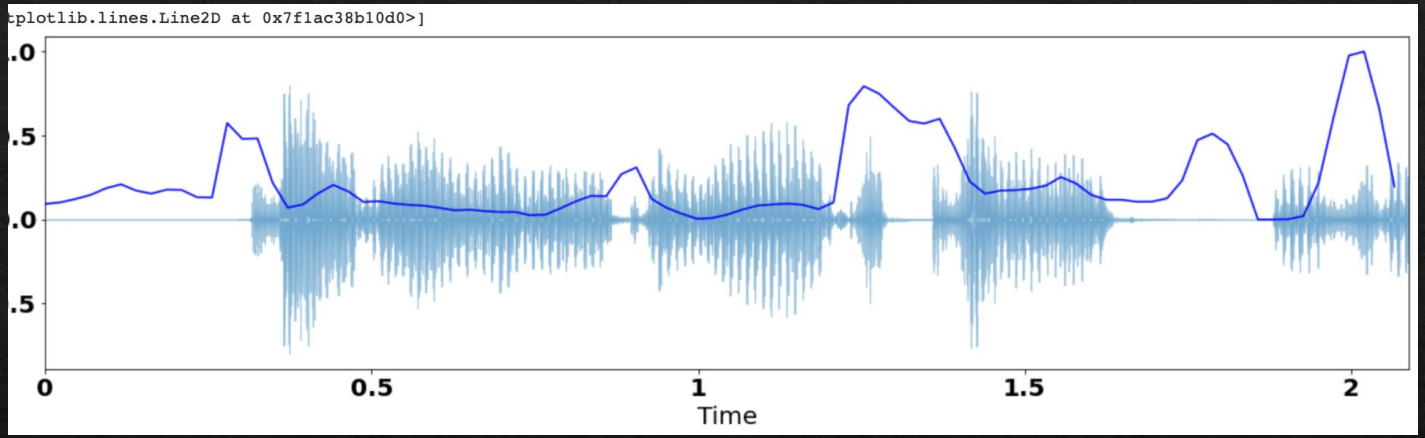
Spectral Centroid

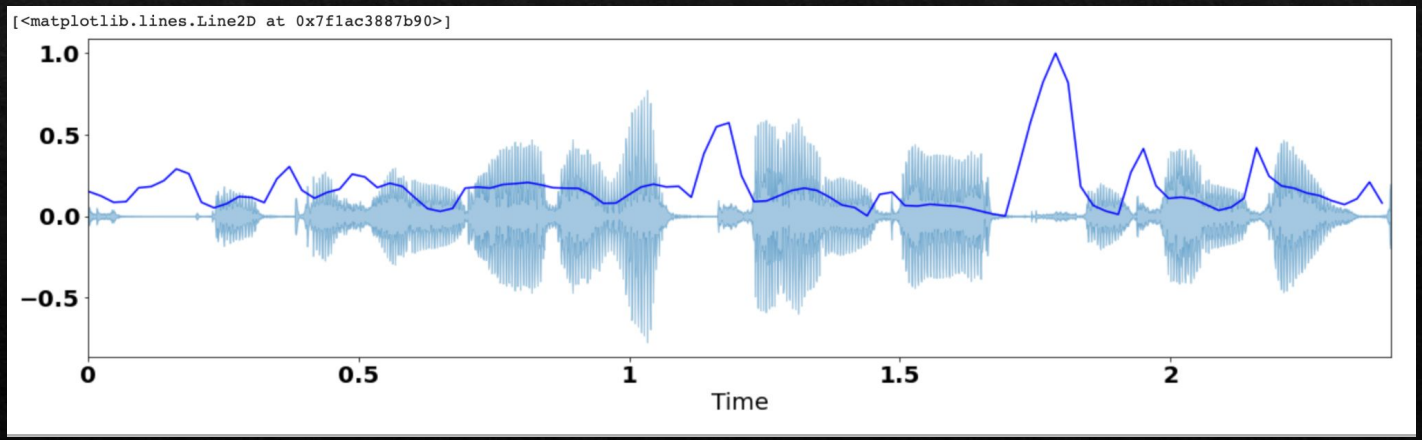
Spectral Centroid

- Where the **center of mass** of the spectrum is located
- Characterizes brightness or perceived quality or **timbre** of sound
- *Timbre* makes a particular musical instrument or human voice have a different sound from another

```
# Spectral centroid - male voice

import sklearn
spectral_centroids = librosa.feature.spectral_centroid(y_m, sr=sr_m)[0]
# Computing the time variable for visualization
plt.figure(figsize=(20, 5))
frames = range(len(spectral_centroids))
t_m = librosa.frames_to_time(frames)
# Normalising the spectral centroid for visualisation
def normalize(x, axis=0):
    return sklearn.preprocessing.minmax_scale(x, axis=axis)
#Plotting the Spectral Centroid along the waveform
librosa.display.waveplot(y_m, sr=sr_m, alpha=0.4)
plt.plot(t_m, normalize(spectral_centroids), color='b')
```

Spectral Centroid





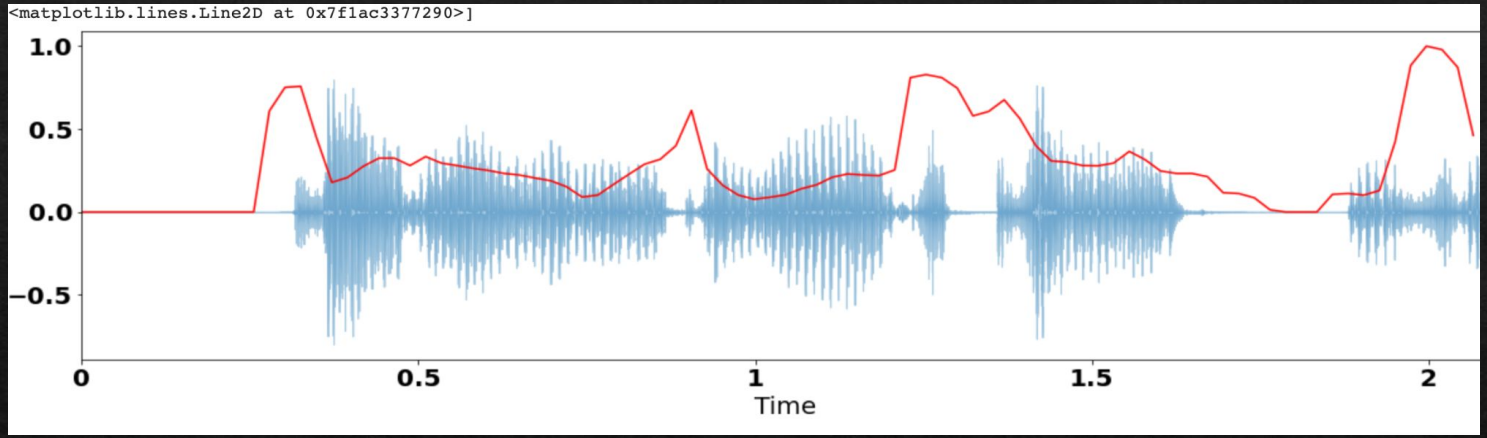
Spectral Rolloff

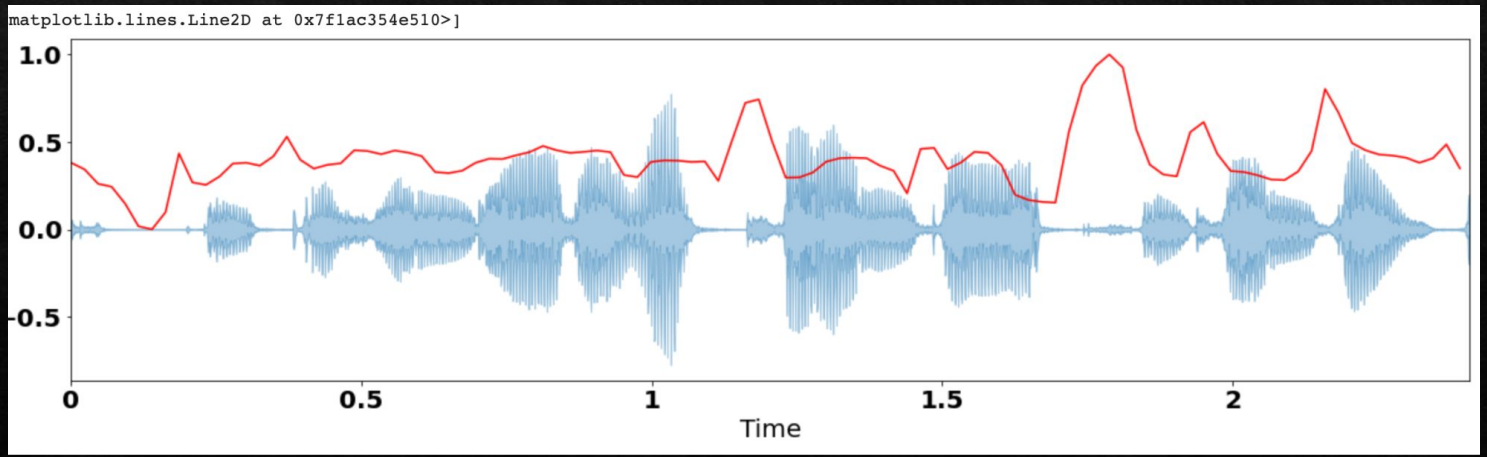
Spectral Rolloff

- **Frequency below which 85% of the total spectral energy or magnitude distribution is concentrated**

```
# Spectral Rolloff - male voice
spectral_rolloff = librosa.feature.spectral_rolloff(y_m+0.01, sr=sr_m)[0]
plt.figure(figsize=(20, 5))
librosa.display.waveplot(y_m, sr=sr_m, alpha=0.4)
plt.plot(t_m, normalize(spectral_rolloff), color='r')
```

Spectral Rolloff





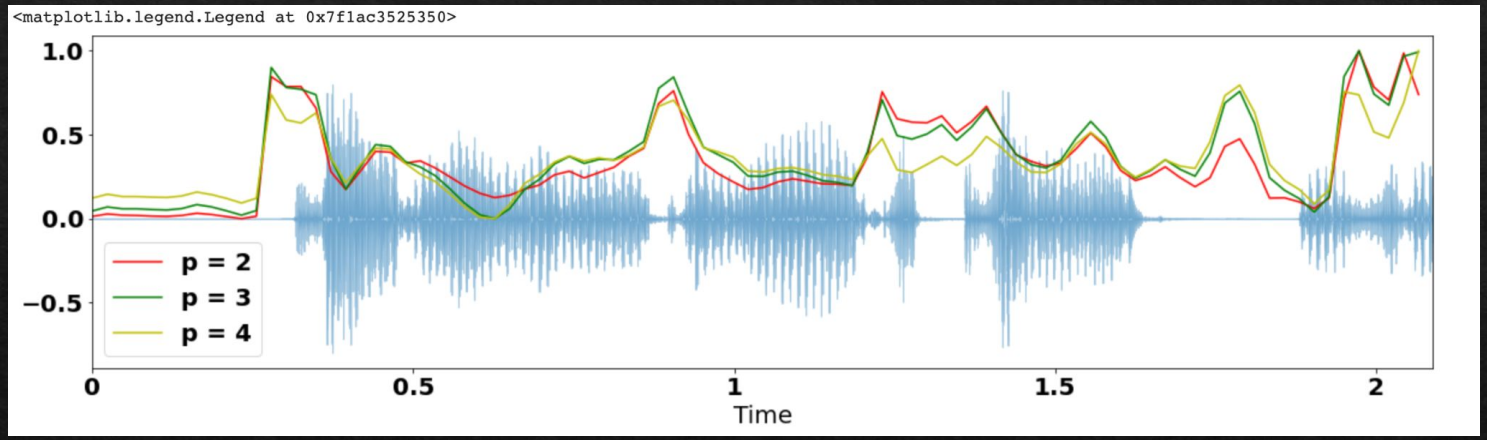
Spectral Bandwidth

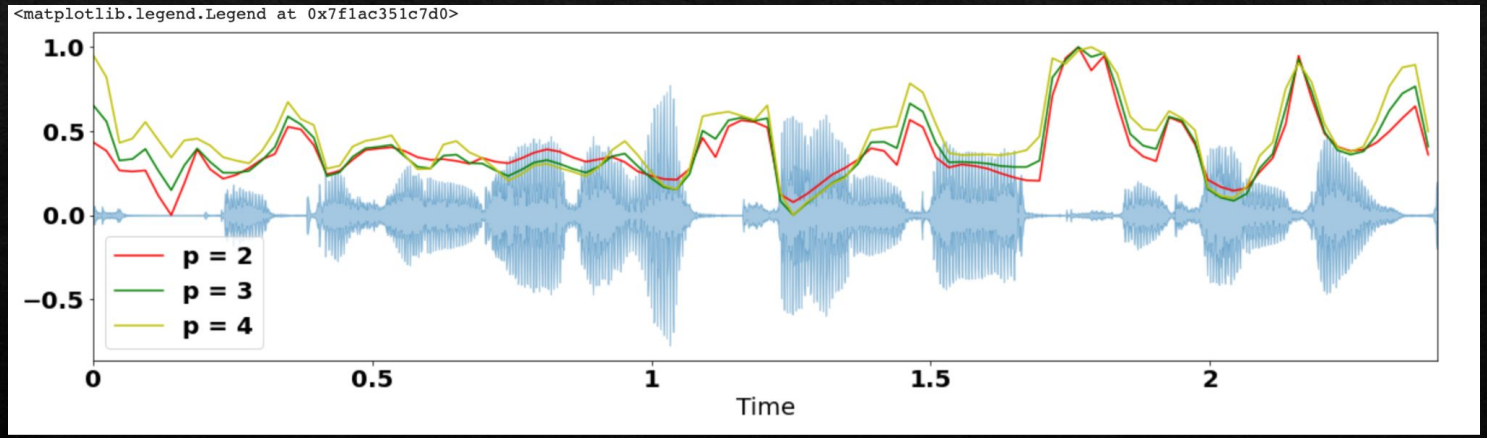
Spectral Bandwidth

- **Defines boundaries (cutoffs) in the spectrum outside which energy flowing through the system begins to be reduced**

```
# Spectral Bandwidth - male voice
spectral_bandwidth_2 = librosa.feature.spectral_bandwidth(y_m+0.01, sr=sr_m)[0]
spectral_bandwidth_3 = librosa.feature.spectral_bandwidth(y_m+0.01, sr=sr_m, p=3)[0]
spectral_bandwidth_4 = librosa.feature.spectral_bandwidth(y_m+0.01, sr=sr_m, p=4)[0]
plt.figure(figsize=(20, 5))
librosa.display.waveplot(y_m, sr=sr_m, alpha=0.4)
plt.plot(t_m, normalize(spectral_bandwidth_2), color='r')
plt.plot(t_m, normalize(spectral_bandwidth_3), color='g')
plt.plot(t_m, normalize(spectral_bandwidth_4), color='y')
plt.legend(('p = 2', 'p = 3', 'p = 4'))
```

Spectral Bandwidth





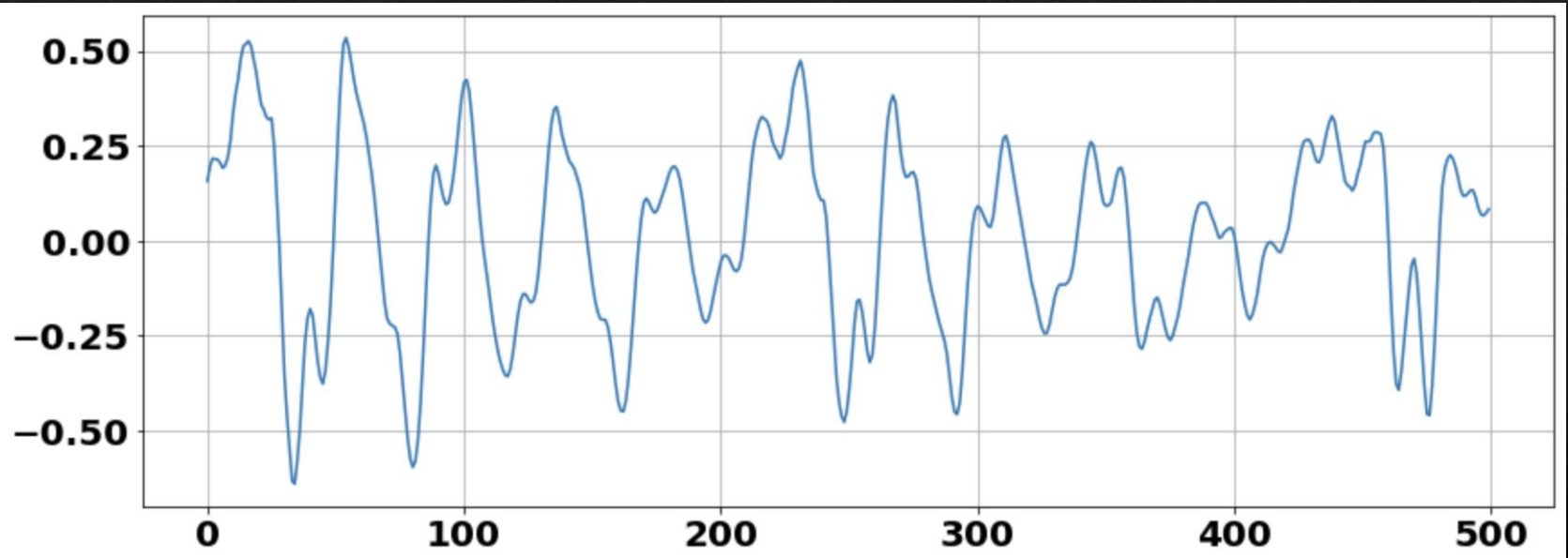
Zero Crossing Rate

Zero Crossing Rate

- How many times signal crosses 0 within a specified time interval

```
#zero-crossing rate - male voice

#Plot the signal:
plt.figure(figsize=(20, 5))
librosa.display.waveplot(y_m, sr=sr_m)
# Zooming in
n0 = 9000
n1 = 9500
plt.figure(figsize=(14, 5))
plt.plot(y_m[n0:n1])
plt.grid()
```

Zero Crossing Rate




```
zero_crossings_m = librosa.zero_crossings(y_m[n0:n1], pad=False)
print(sum(zero_crossings_m))
```

```
22
```

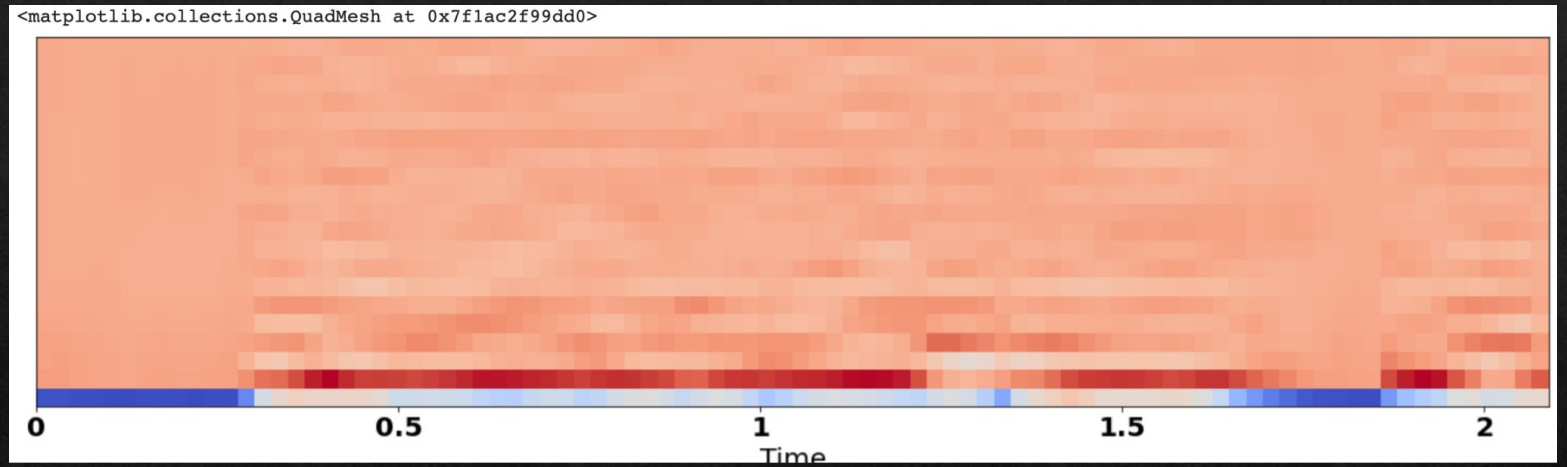
MFCCs

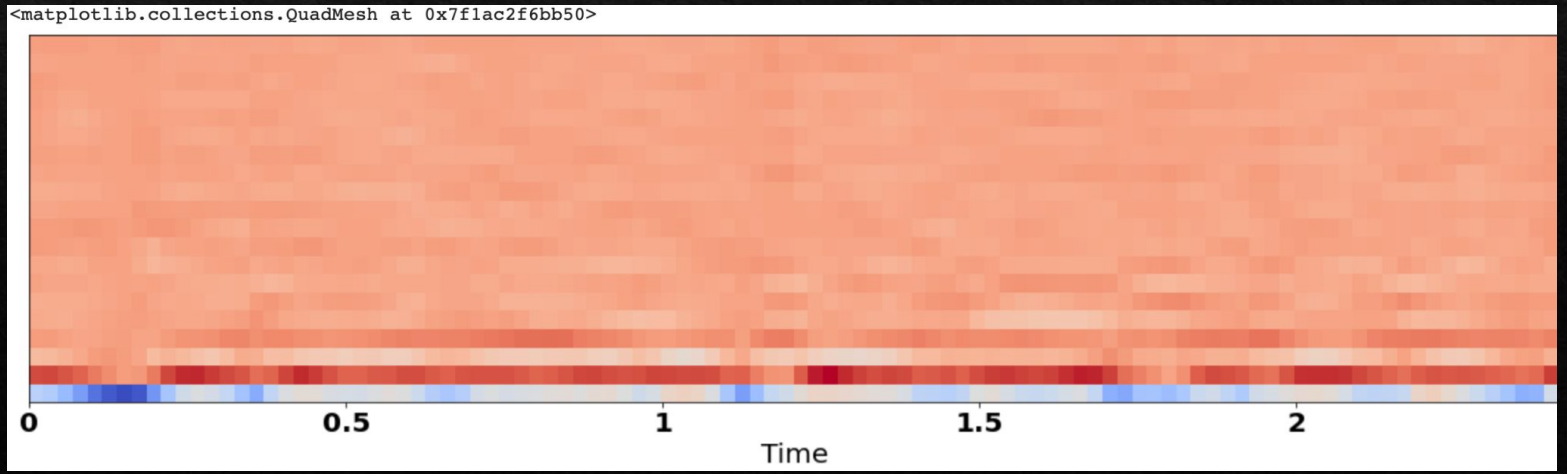
Mel-frequency cepstral coefficients (MFCCs)

- Characterize **mel-frequency cepstrum**
- *Mel Scale* - approximates human auditory system's response more closely
- *Cepstrum* - explores periodic structures in frequency spectrum (spectrum of spectrum)
- *MFC* - frequency bands are equally spaced on the mel scale
  - As opposed to linearly-spaced frequency bands used in the normal spectrum

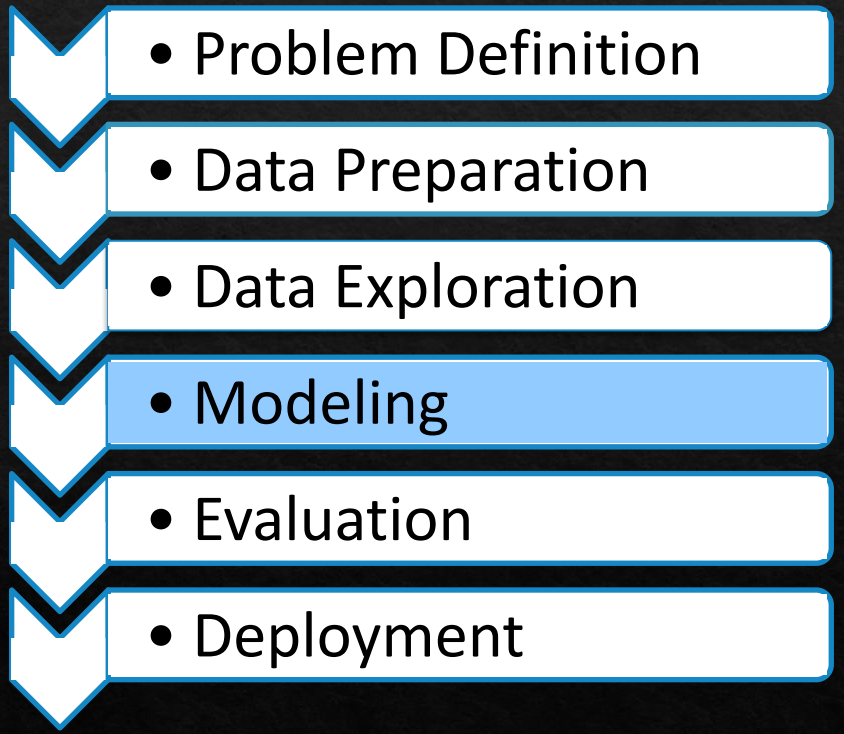
```
#mfcc - male voice
mfccs_m = librosa.feature.mfcc(y_m, sr=sr_m)
#Displaying  the MFCCs:
plt.figure(figsize=(20, 5))
librosa.display.specshow(mfccs_m, sr=sr_m, x_axis='time')
```

MFCCs


<matplotlib.collections.QuadMesh at 0x7f1ac2f99dd0>
0
0.5
1
1.5
2
Time

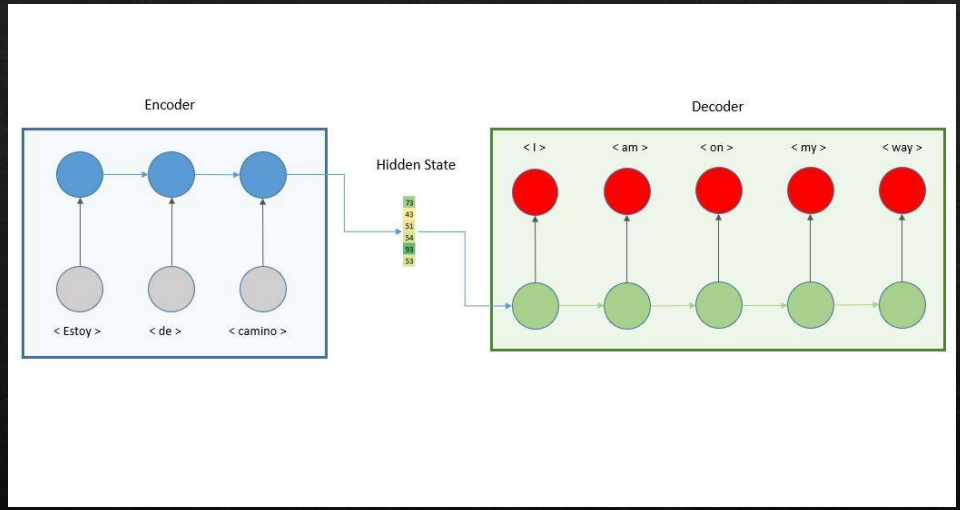

<matplotlib.collections.QuadMesh at 0x7f1ac2f6bb50>
0
0.5
1
1.5
2
Time

Data Mining - 6 steps



Problem Type

Sequence-to-sequence learning Seq2Seq



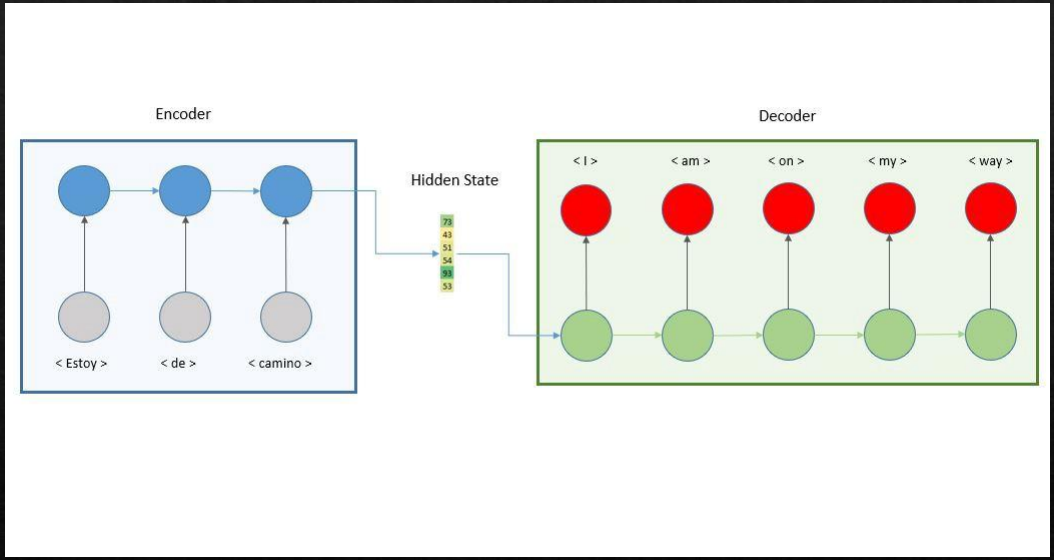
inputs **: sequences in one domain**
outputs **: sequences in another domain**

- **Machine Translation : SPANISH text -> ENGLISH text**
- **Video Captioning : video (images) -> text**
- **Speech Recognition : audio -> text**

Model Setup

Encoder - Decoder Network

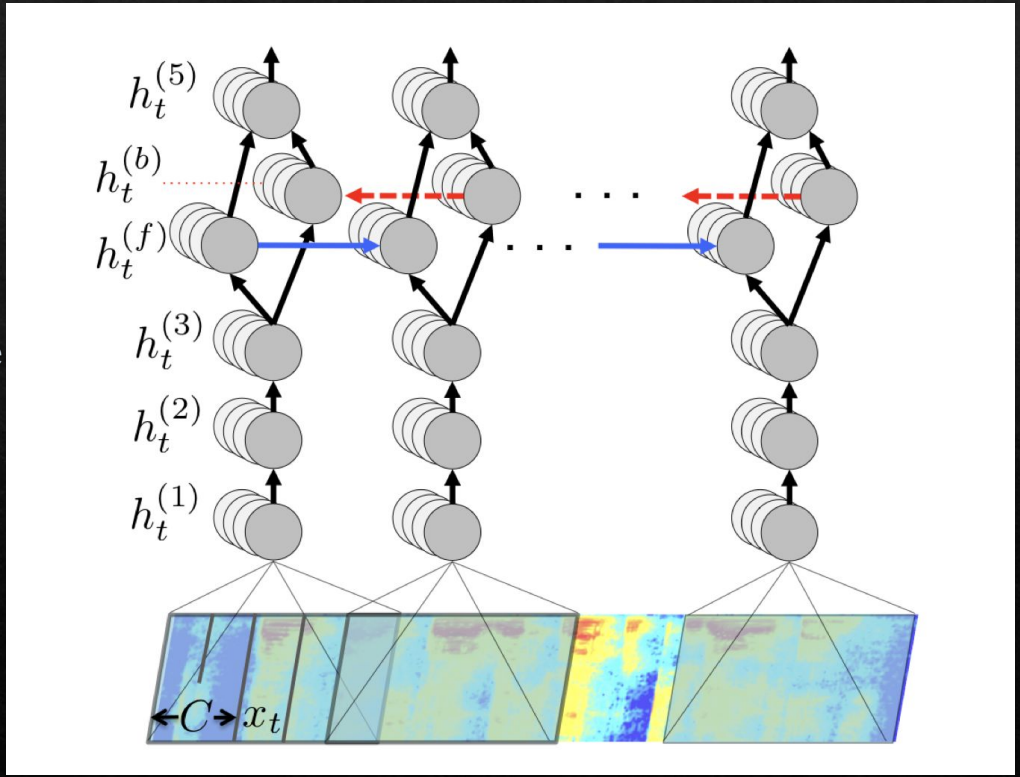
- ENCODER
  - Learn representation of input
  - Context and temporal dependencies need to be preserved
- DECODER
  - Learn to construct another sequence from the encodings
  - Probability distribution over entire vocabulary of tokens



Model Architecture

Paper -
Deep Speech: Scaling up end-to-end speech recognition, 2014

- (X -> audio, Y -> text) pair
- Input - Spectrogram(X)
  - Every time slice denotes the power of the p'th frequency bin in the audio frame at time t
- Output
  - Sequence of character probabilities for the transcription y, with $\hat{y}_t = P(c_t|x)$, where $c_t \in \{a, b, c, \ldots, z, \text{space}, \text{apostrophe}, \text{blank}\}$



Model Architecture

Paper -
Deep Speech 2: End-to-End Speech Recognition in English and Mandarin, 2015

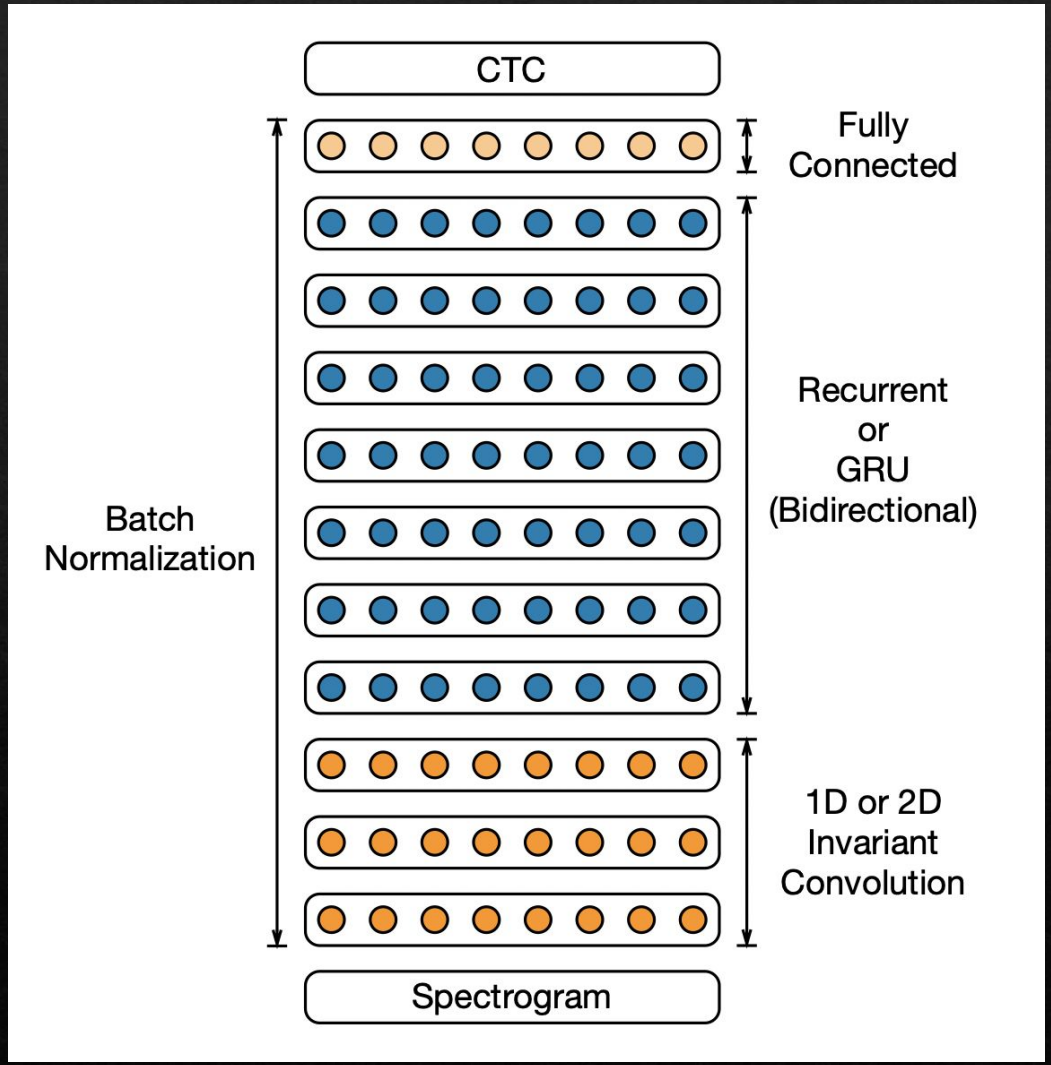
- **1-3 CONV layers**

$$h_{t,i}^{l} = f(w_i^l \circ h_{t-c:t+c}^{l-1})$$

- **1-7 GRU layers**

$$\overrightarrow{h}_t^l = f(W^l h_t^{l-1} + \overrightarrow{U}^l \overrightarrow{h}_{t-1}^l + b^l)$$

$$\overrightarrow{h}_t^l = g(h_t^{l-1}, \overrightarrow{h}_{t-1}^l)$$
$$\overleftarrow{h}_t^l = g(h_t^{l-1}, \overleftarrow{h}_{t+1}^l)$$



Model Implementation

- Build a Vocabulary

```
Vocab size: 30 (with BLNK, PAD, UNK and SPACE added)
vocab[0]: <blank>
vocab[1]: <pad>
vocab[2]: <unk>
vocab[3]:
```

Encoder

- Read audio file
- Compute Mel Spectrogram
- Create tensor of shape (seq_len, batch, in_features) over all audio files

```
class Encoder(nn.Module):
  def __init__(self):
    super().__init__()
    self.spec = torchaudio.transforms.MelSpectrogram(sample_rate=16000, n_mels=80)

  def forward(self, input_X):
    X_len = []
    AUDIO = []
    for X in input_X:
      waveform, sample_rate = torchaudio.load(io.BytesIO(X))
      audio_tensor = self.spec(waveform).squeeze(0).transpose(1, 0)
      # print('spectrogram', audio_tensor.shape)
      AUDIO.append(audio_tensor)
      X_len.append(audio_tensor.shape[0])
    AUDIO = pad_sequence(AUDIO, padding_value=1.)
    # print('encoder', AUDIO.shape)
    # print(X_len)
    return AUDIO, X_len
```

Model Implementation

Decoder

- 2 CONV, 3 GRU, 1 Linear
- Input : (in_seq_len, batch, in_features)
- Output : (out_seq_len, batch, vocab_size)

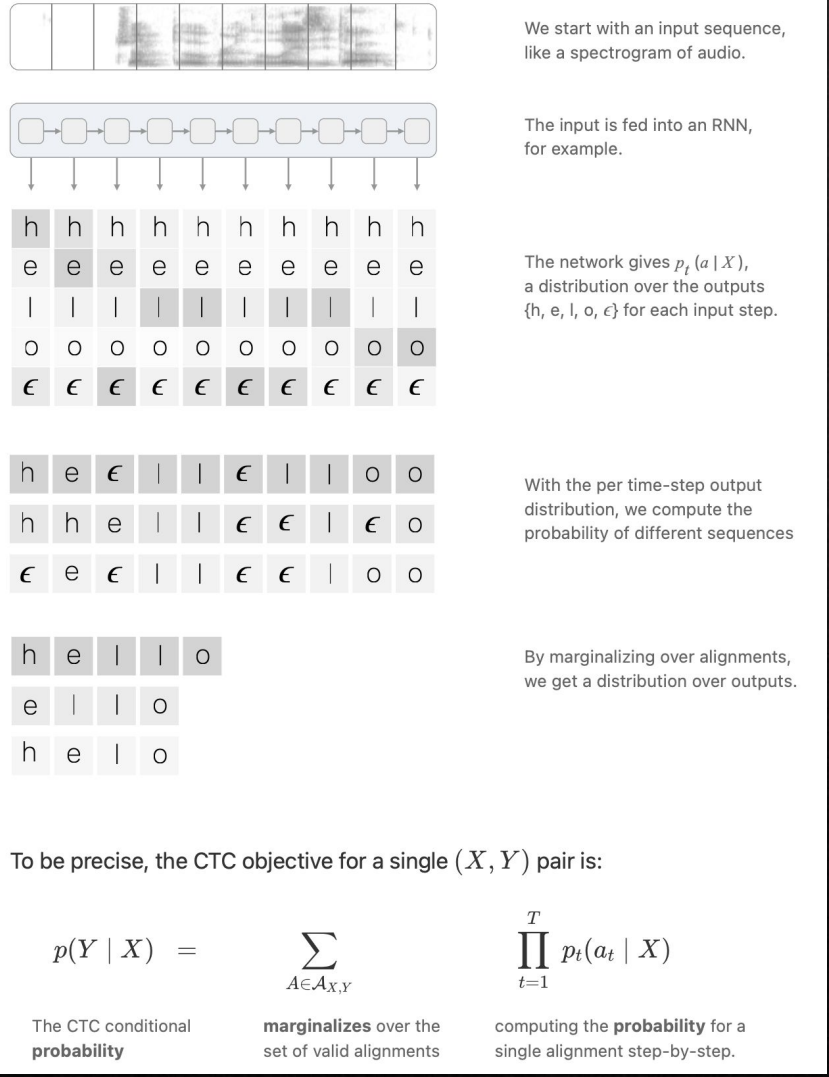
```
Decoder(
  (conv): Sequential(
    (0): Conv2d(1, 32, kernel_size=(5, 5), stride=(1, 1), padding=same)
    (1): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): Hardtanh(min_val=0, max_val=20.0, inplace=True)
    (3): Conv2d(32, 32, kernel_size=(5, 5), stride=(1, 1), padding=same)
    (4): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (5): Hardtanh(min_val=0, max_val=20.0, inplace=True)
  )
  (rnns): Sequential(
    (0): RNNWrapper(
      (rnn): GRU(2560, 105, bias=False, bidirectional=True)
    )
    (1): RNNWrapper(
      (batch_norm): OverLastDim(
        (module): BatchNorm1d(105, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      )
      (rnn): GRU(105, 105, bias=False, bidirectional=True)
    )
    (2): RNNWrapper(
      (batch_norm): OverLastDim(
        (module): BatchNorm1d(105, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      )
      (rnn): GRU(105, 105, bias=False, bidirectional=True)
    )
  )
  (fc): OverLastDim(
    (module): Sequential(
      (0): BatchNorm1d(105, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (1): Linear(in_features=105, out_features=30, bias=False)
    )
  )
)
```

Training

Connectionist Temporal Classification (CTC) Loss

```
out = self.fc(x)
logprobs = nn.functional.log_softmax(out, dim=2)
# print('decoder', logprobs.shape)

# compute CTC loss
ctc_loss = nn.CTCLoss(zero_infinity=True)
loss = ctc_loss(logprobs, Y_in, X_len, Y_len)
return loss
```

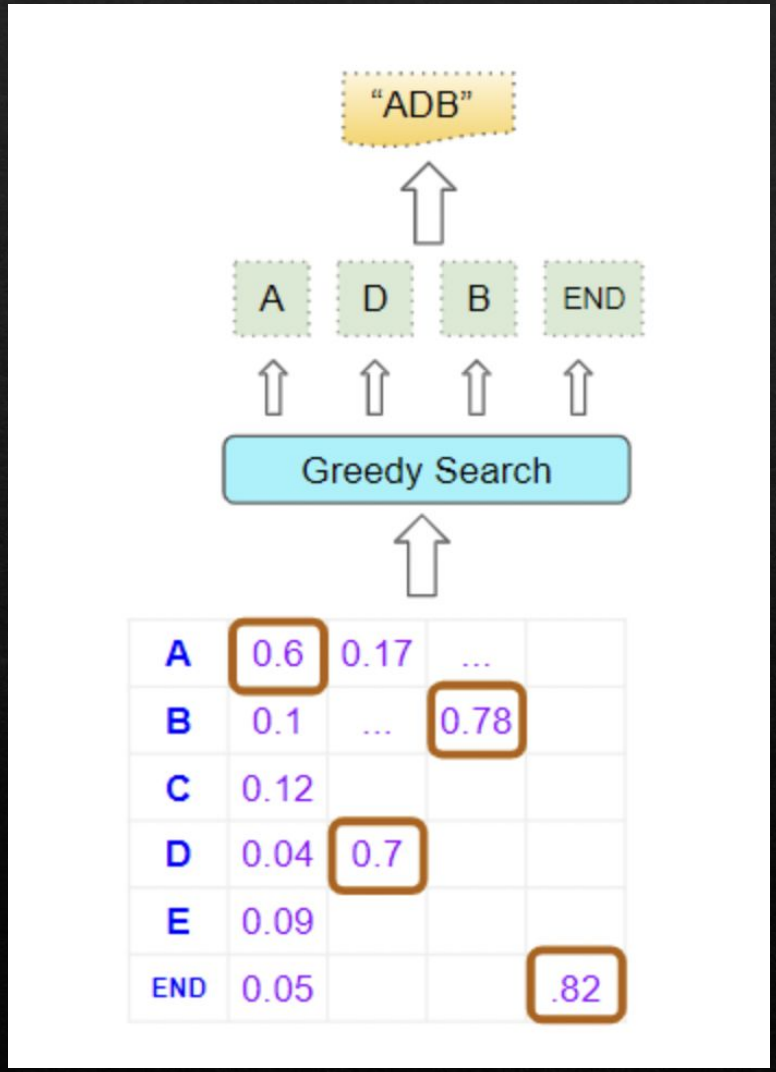
- Adam optimizer
- Back-propagate the gradient to update the model parameters
- Model checkpointing



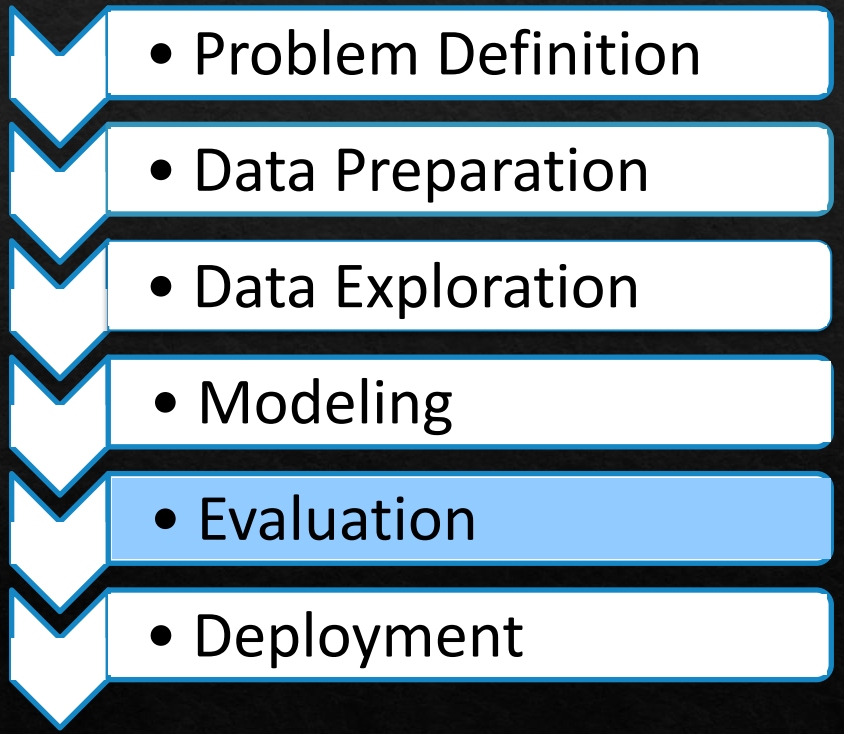
Inference

Greedy Search

- Take just the single best word/character at each position
- Remove <pad> and <blank> tokens
- Considers each position in isolation
  - without examining what came before
  - another approach - beam search



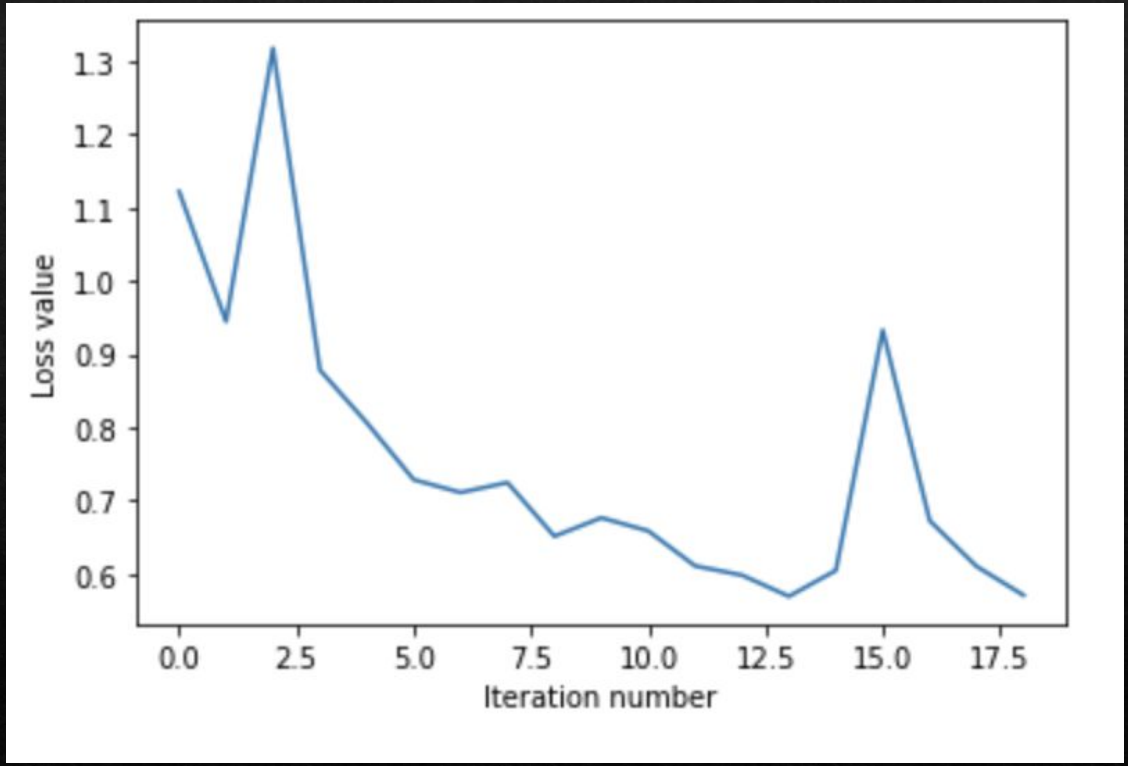
Data Mining - 6 steps



Model Tuning / Validation set results

Hyperparameters

- batch_size = 32
- load_size = 1280
  - Total training load = 856,210 ~ 669 x 1280
- n_mels = 80
- num_epochs = 20
- learning_rate = 0.0006



Validation set results

Evaluation on validation set

- 53,401 audio segments

```
Avg. Levenshtein Distance on validation set 12.748169562612343
Avg. Character Error Rate on validation set 0.353506446877744626
Avg. Word Error Rate on validation set 0.7240277065417712
```

Validation set results

actual	predicted
ask simple even dumb questions all to	ask simple even dum questions all
um and i dont think in some cases	um and i dot think in somehases
maybe thats okay like some but but this	maybe thats okay like some but wut th
because we didnt have market exposure	because we didnnt have morket explosinan
correct me if im wrong but you know	correctn if fom rong but you know
around the whole thing do you think it	aroung the whole thing do you think
um i think its underrated how how	um i think its under rated hom paw
thoughts about his interaction with elon	thouhts about his interaction wit e line
like no were not being ageist here	like kno wer not being agust her
and paying for them but just being	ind paing for them but just beng
real and then moments like this that was	rel and then moments like this ta
worse by the way he presents himself	wors by the way he present somself
possibly ever handle theres a bunch of	possibly ever handal theres a bunc
so when you get when you look at	so when you get when you loke
every stock has one year momentum how	every stak has one year amenten ha
is looking at the data after all this is	looking at the data after all of this is
um i guess one way to say that is the	um i guess one way to say thas
but the higher the stakes the more the	i the hire the stakes the more
but um what you want to do is actually	ot um what you want to doi is actually
a large amount of patterns that appeared	a larg amout of patterans that a peoe
say that comes to mind that	say thi comes ti mind that
that then runs this model and takes	that then runs tis smodel and takes
are like is this like pay to play right	like is tis like pate a play right
of flavor of community and	if flawer of commuly an

Test set results

Evaluation on test set

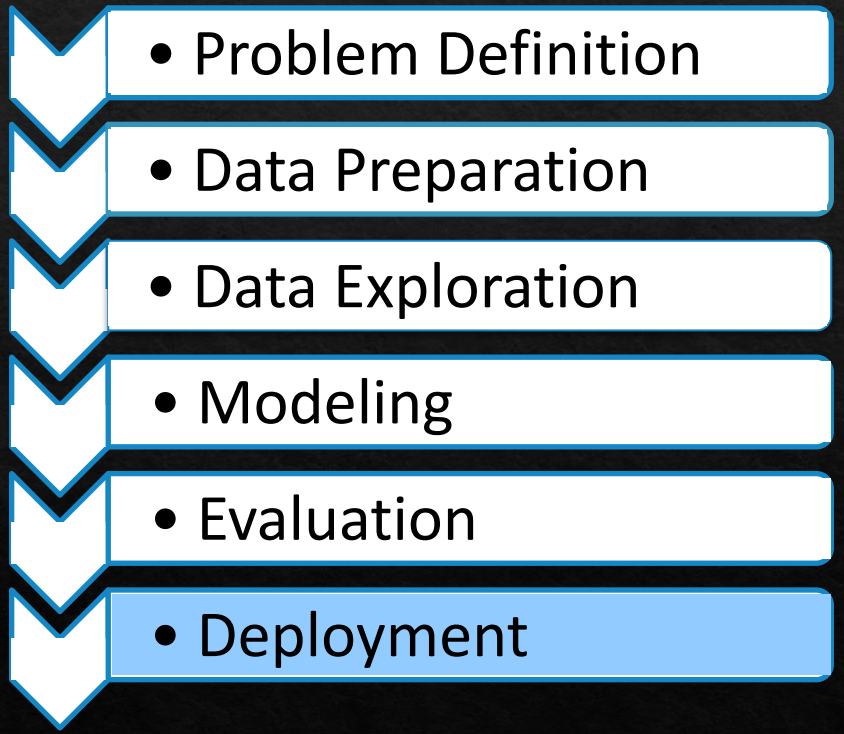
- **Note - Only run once with the final model saved after tuning.**
- **53,468 audio segments**

```
Avg. Levenshtein Distance on test set 12.745196417884927
Avg. Character Error Rate on test set 0.35310290516075804
Avg. Word Error Rate on test set 0.72458834360025
```

Test set results

actual	predicted
yeah sincerity as well like if you if	yes senserity is well like i you
they keep raising funds so they can	they keep rasing funs o they cn
to huge data collection i mean um	to huse deta collection an u
in another world of mine	en another world the mind
and like they all do a ton of	and like it al ll do shit tan
going all in and im like	going al land and um like
like prove me wrong yeah thats	like pove me wrong yeah that
quickly in some way theyre like this is	puickly and some way hea like this
thats one of the most so uh	thats ne of the most so uh
using this methodology of money	using this methadology f money
data science and ai this is uh	data sieence and ai this isuh
i really what i was reading zero to one	i really what i was readings terei ta one
kind of thinking big and	kind of thinking began
optimistic it puts you everything you	uptimisic it puts you everything
really long time on it	really long fine on i
i had a job as a quant	i had a job as a cuont
like a bunch of papers on quant finance	like a bunch of papers on quanfindan
its like its amazing how much how kind	its like its bemazing hom mi how kind
that too many people that know like	that to many people that know like
to capture but its fun to listen to	to cacture but its funto lisens
and youre able to uh explain it so	and youre able to uh explaines
machine learning company blinkist app	machin learning company lnkis ap
thing and then as that develops	thing and then as that develoops
of just or politics you know the the	af just or potics you know o the

Data Mining - 6 steps



Model as a service

- Final model weights downloaded to localhost as .pt file
- Used in a Python script to make inference on a *new* podcast episode (not used in modeling)
  - Input - Youtube video URL
  - Output - Text transcript file
- Another possibility - embed model into a RESTful app

Key Challenges

- Effectively capturing speech characteristics from audio
- Hyperparameter search for a huge training set / long train time
- Limited GPU quota and RAM for more complex model training
- Setting up an end-to-end data science workflow for speech
  - How to maintain / re-train the model for new data?

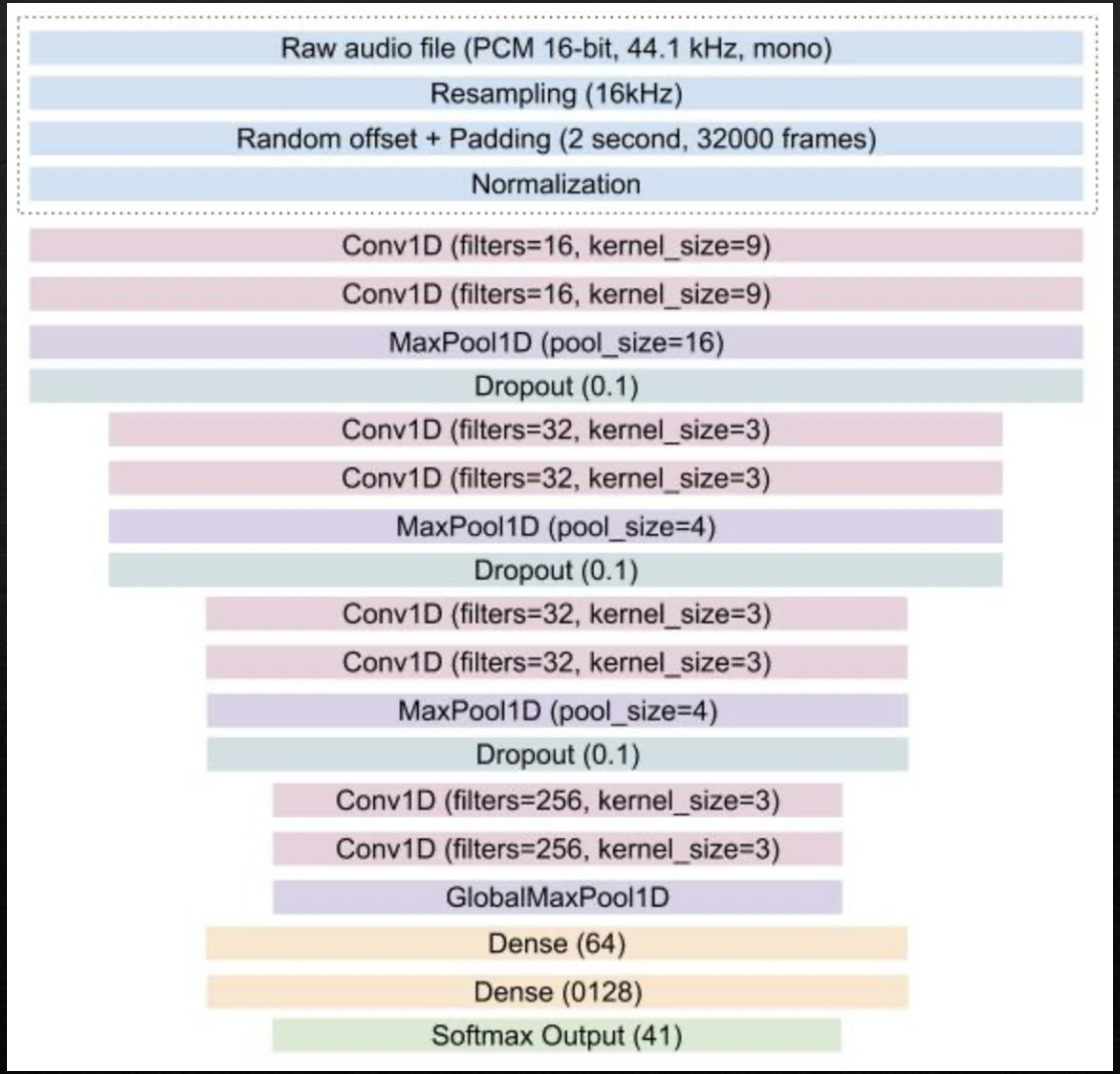
Scope for improvement

- Enriched dataset (raw or augmented)
- More recurrent layers
- More training epochs
- Options for training Encoder / Decoder
  - Encoders - convolutional, transformers
  - Decoders - Other RNN variants, transformers
- Beam search with language model as decoder for inference

Using an explicit Encoder

Automated Feature Extraction

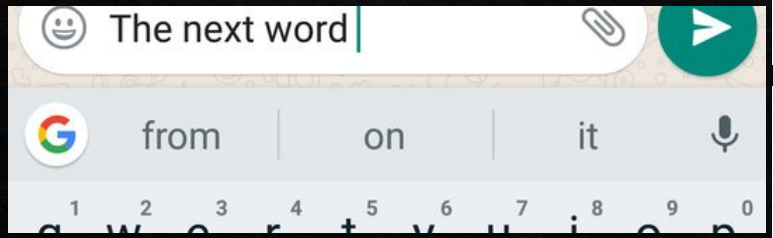
- **Black-box**
- **Little explainability**



Inference / Decoding

Language Model

- Assign a probability to any sequence of words



A fixed-window neural Language Model

output distribution

$$\hat{\boldsymbol{y}} = \text{softmax}(\boldsymbol{U}\boldsymbol{h} + \boldsymbol{b}_2) \in \mathbb{R}^{|V|}$$

hidden layer

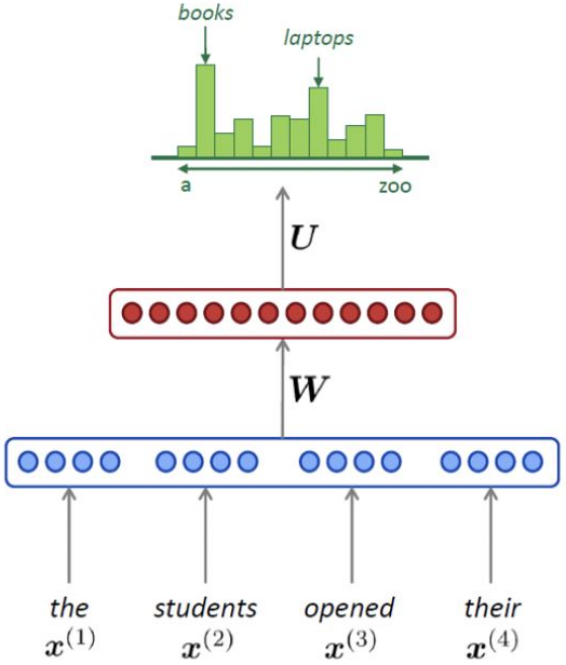
$$\boldsymbol{h} = f(\boldsymbol{W}\boldsymbol{e} + \boldsymbol{b}_1)$$

concatenated word embeddings

$$\boldsymbol{e} = [\boldsymbol{e}^{(1)}; \boldsymbol{e}^{(2)}; \boldsymbol{e}^{(3)}; \boldsymbol{e}^{(4)}]$$

words / one-hot vectors

$$\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \boldsymbol{x}^{(3)}, \boldsymbol{x}^{(4)}$$

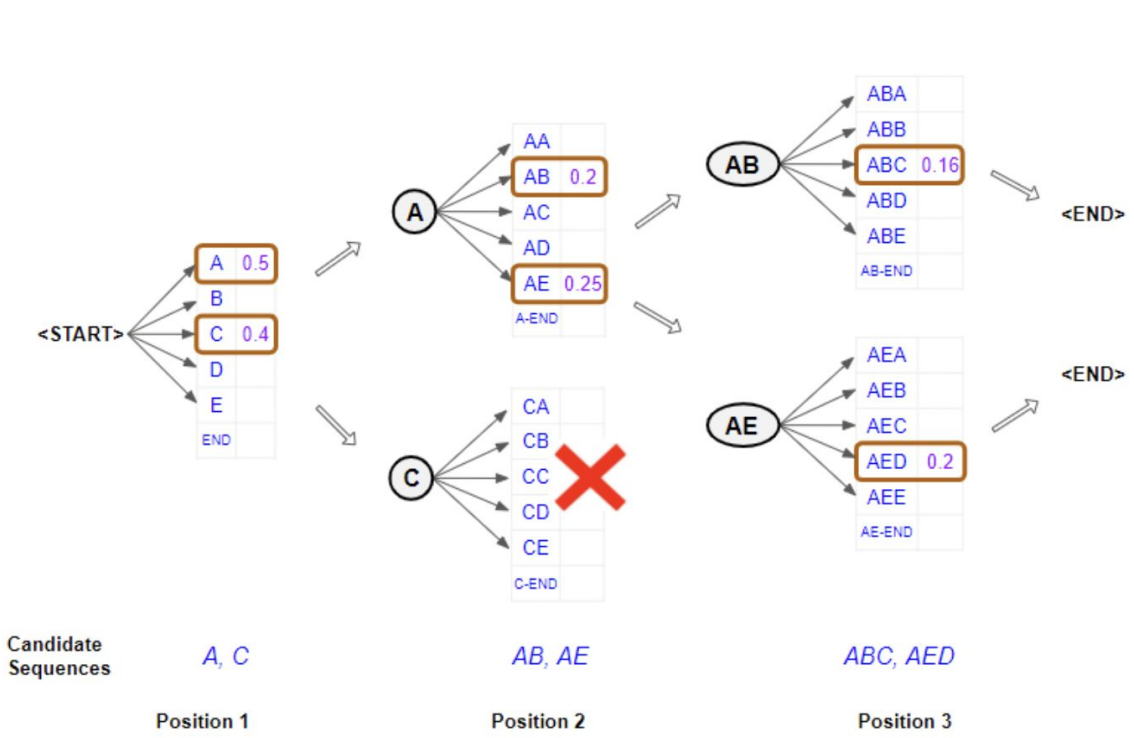


$$P(w_i|w_1, w_2, \ldots, w_{i-1}) = \frac{P(w_1, w_2, \ldots, w_i)}{P(w_1, w_2, \ldots, w_{i-1})}$$

Inference / Decoding

Beam Search

- Set beam width = B
- Take the best ‘B’ words
- Picks ‘B’ best sequences so far
  - Combination of all preceding words and word at the current position



Beam Search example, with width = 2 (Image by Author)

Predict the one with the higher overall probability

Inference / Decoding

Example from Machine Translation

- **Beam width = 3**

Hindi input sequence -> मैं भारत में अपने माता-पिता से मिलने जा रही हूं।
English probable output sequence -> I am going to meet my parents in India → 0.56
I am visiting my parents in India → 0.49
I will visit India to meet my parents → 0.39

Three output sentences with the highest conditional probabilities and different lengths

Inference with beam search

Paper -
Deep Speech: Scaling up end-to-end speech recognition, 2014

- **Beam width in the range 1000-8000**

RNN output	Decoded Transcription
what is the weather like in bostin right now	what is the weather like in boston right now
prime miniter nerenr modi	prime minister narendra modi
arther n tickets for the game	are there any tickets for the game

Table 1: Examples of transcriptions directly from the RNN (left) with errors that are fixed by addition of a language model (right).

References

- Dataset - https://www.youtube.com/watch?v=ICj8p5jPd3Y&list=PLrAXtmErZgOdP_8GztsuKi9nrraNbKKp4

- Audio Exploration
  - https://www.kaggle.com/oakmin/audio-data-exploration
  - https://www.kaggle.com/fizzbuzz/beginner-s-guide-to-audio-data
  - https://www.kaggle.com/oakmin/audio-data-exploration

- Seq2Seq - https://towardsdatascience.com/what-is-an-encoder-decoder-model-86b3d57c5e1a

References

- ASR - https://towardsdatascience.com/customer-case-study-building-an-end-to-end-speech-recognition-model-in-pytorch-with-assemblyai-473030e47c7c

- ASR Evaluation https://symbl.ai/blog/key-metrics-and-data-for-evaluating-speech-recognition-software/

- WER - https://www.assemblyai.com/blog/word-error-rate/

- Model Architectures
  - https://arxiv.org/pdf/1412.5567.pdf
  - https://arxiv.org/pdf/1512.02595.pdf

THANK YOU