HELLO WHISPY!

SHREDDING REAL-TIME AUDIO WITH STT MODELS

HELLO WHISPY!

SHREDDING REAL-TIME AUDIO WITH STT MODELS

NSFW: live presentation

HELLO WHISPY!

SHREDDING REAL-TIME AUDIO WITH STT MODELS

🚨🚨

NSFW: live presentation

FIRST THINGS FIRST

BIG ROUND OF THANK YOUs

TO ALL ATTENDEES! THAT'S YOU!

Thank you for showing up & making JanusCon awesome!

SO... HELLO!

ANTONIO BEVILACQUA

(yes, I play catpipes)

ML ENGINEER @MEETECHO

A SAD, SAD STORY

A SAD, SAD STORY

I LEFT MY CLICKER

IN IRELAND

A SAD, SAD STORY

IT WAS

BEAUTIFUL

RECOVERY PLAN

RECOVERY PLAN

1. Getting hired by tech company

RECOVERY PLAN

1. Getting hired by tech company
2. Build STT system for real-time audio

RECOVERY PLAN

1. Getting hired by tech company
2. Build STT system for real-time audio
3. Design voice-activated thingy to simulate mouse/keyboard

RECOVERY PLAN

1. Getting hired by tech company
2. Build STT system for real-time audio
3. Design voice-activated thingy to simulate mouse/keyboard
4. Wait for JanusCon

RECOVERY PLAN

1. Getting hired by tech company
2. Build STT system for real-time audio
3. Design voice-activated thingy to simulate mouse/keyboard
4. Wait for JanusCon
5. Present it without using (much) stupid fingers for transitions

RECOVERY

6. FAIL

/keyboard

uch) stupid fingers for transitions

TODAY IS THE DAY!

DID YOU SAY PIZZA?

LET'S START!

STT

Use spoken-language auditory data for cool things like:

Transcription

Translation

Voice Activity Detection

Language Identification

WHISPER

Family of multilingual, multitask ASR models

Powered by OpenAI 

Trained on 600k+ hours of audio

Domain SOTA (for zero-shot learning)

Several flavours & toppings available

TERRIFIC, BUT...

Whisper is limited to offline use only

Whisper models occasionally hallucinate (sounds fun but it's not)

ABOUT WHISPY

WHAT IS WHISPY?

A self-contained, production-ready transcription service

Based on extensible pipeline system for real-time AI applications

The latest inclusion to our Meetecho products!

BASIC IDEA

Read in streaming audio from source and buffer it

Transcribe audio chunks

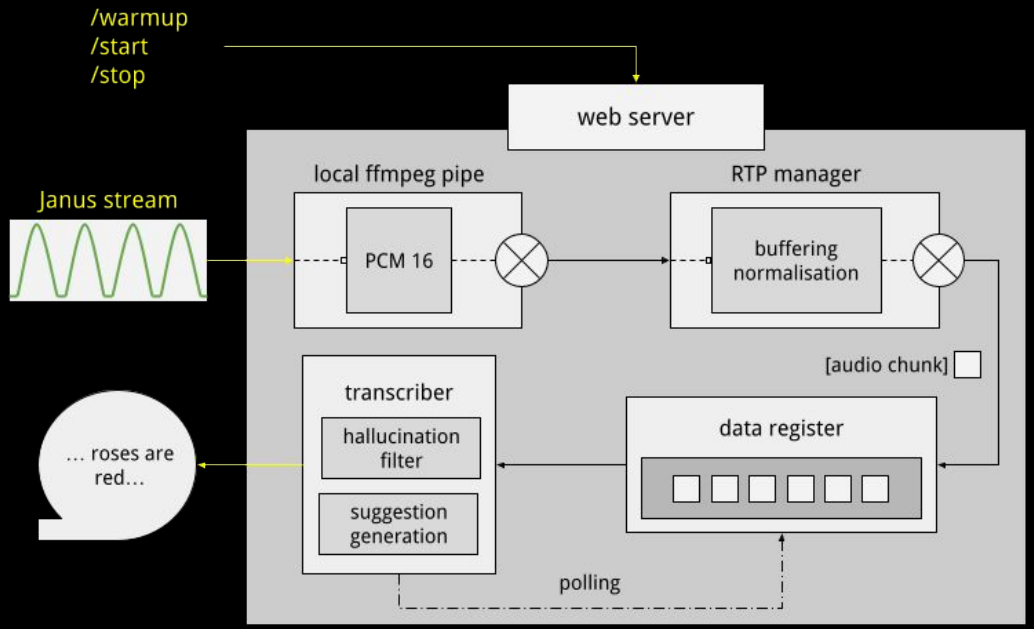
Detect hallucinations & other unwanted artifacts

Merge transcriptions after filtering out overlapping parts

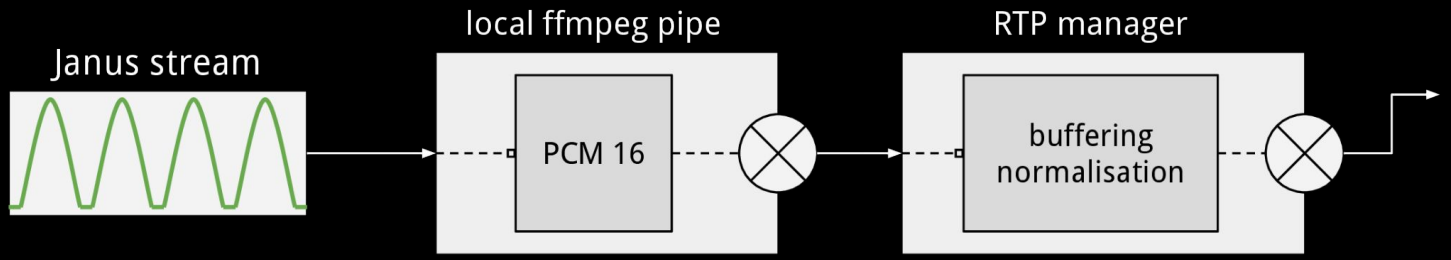
MAIN INGREDIENTS

1. Input pipeline (`janus`-powered `ffmpeg` stream)
2. Shifting data register
3. Pre-trained STT Whisper model
4. Hallucination filter / suggestion generator

MAIN INGREDIENTS



INPUT PIPELINE



Using `ffmpeg-python`, read in a forwarded Janus remote audio stream and produce a local RTP stream that is consumed by a naive client.

Demand data manipulation (encoding/decoding) to `ffmpeg`, can easily be extended to multiple input/output scenarios.

DATA REGISTER

c_n : audio chunks
c_d : temporal duration of each chunk
f_s : audio sampling rate

buffer max length = c_n * c_d * f_s

from [secs]
to [secs]
seq_num

TL;DR FIFO Q

Keeps track of chunk numbers

Stores temporal coordinates

TRANSCRIBER

Inference happens on consecutive, overlapping audio chunks

Data already come in a Whisper-friendly format (thanks to input pipe)

Trick #1: re-transcription (helps with delays & context)

Trick #2: filter out silent regions (helps with speed & resources)

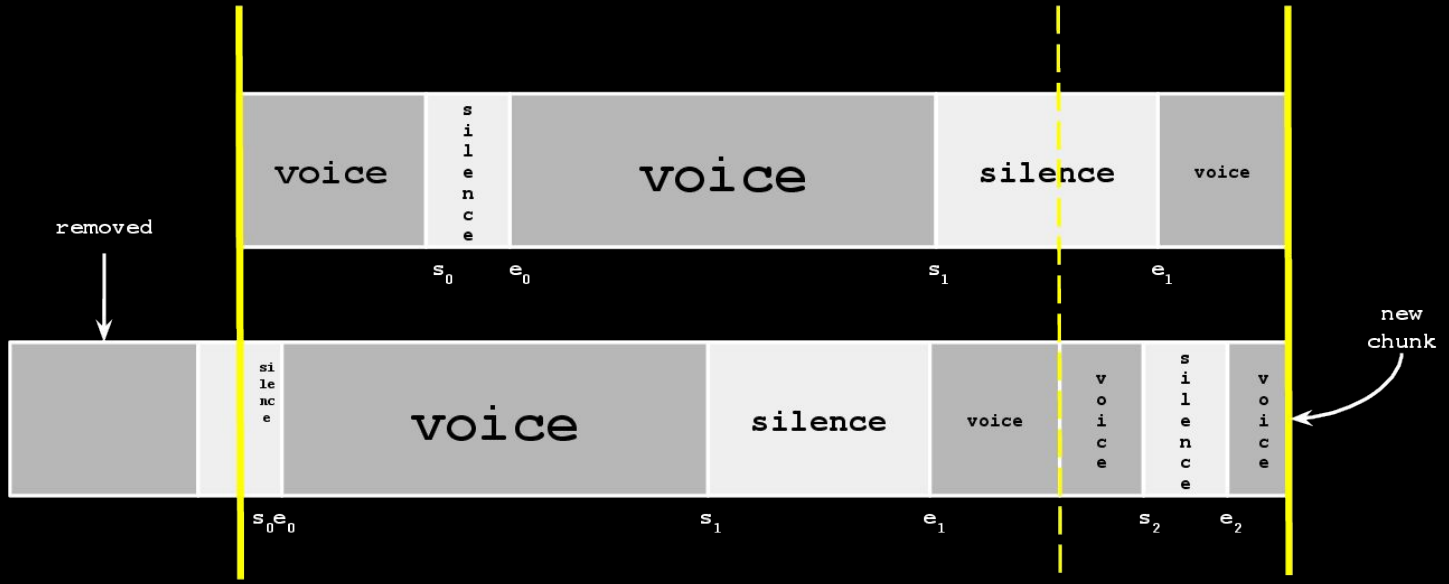
SILENCE FILTER: VAD

Whisper comes with an embedded Voice Activity Detection filter (Silero)

Useful to detect voice segments within audio data

In our chunking system, coordinates need to be stored & shifted as new audio segments are available from the source

SILENCE FILTER: VAD



TRANSCRIBER CONTD.

Not really far from real code!

`model` is `faster-whisper`
(quantised = small & fast)

Next: hallucinations and suggestions!

```
last_chunk = register.get_last_chunk()
voice_timestamps, voice_duration = run_vad(last_chunk)

if voice_duration == 0.0:
    register.flush()
    return 'silence'

coordinates.update(voice_timestamps)

speech = get_speech(register.data, coordinates)
transcript = model.transcribe(speech)
hallucinations = has_hallucination(transcript)

if len(hallucinations) > 0:
    transcript = fix_hallucination(transcript)

transcript = generate_suggestion(transcript, last_transcript)

return transcript
```

HALLUCINATION FILTER

Repeated tokens

Trivial to solve (kinda) as a token is often repeated several times in the same block.

That is a great place to eat **pizza**, **pizza**, **pizza**, **pizza**, **pizza**, **pizza**, **pizza**, **pizza**, **pizza**, **pizza** even without pineapple on it.

Repeated sentences

Trick of the trade: find motifs in sentences using MatrixProfile

(like shooting a fly with a bazooka: effective AND cool)

And neither of these stories is very inspiring or great, but one of them is distinct. **We'll be right back. We'll be right back. We'll be right back. We'll be right back. We'll be right back. We'll be right back. We'll be right back.**

SUGGESTION GENERATION

Treat transcriptions as sequences of words

Find word in current transcript that minimises Levenshtein distance

Use target word as cutting point for current transcript

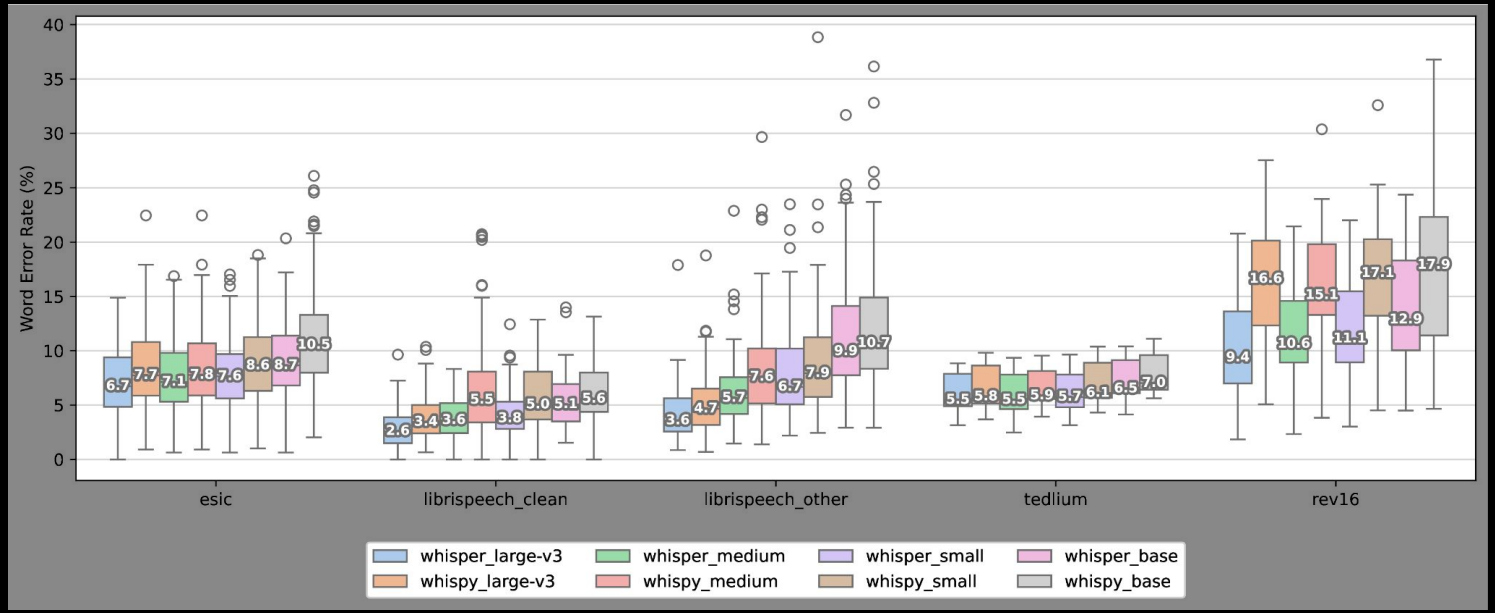
Concatenate suggestion w/ previous transcript

SUGGESTION GENERATION

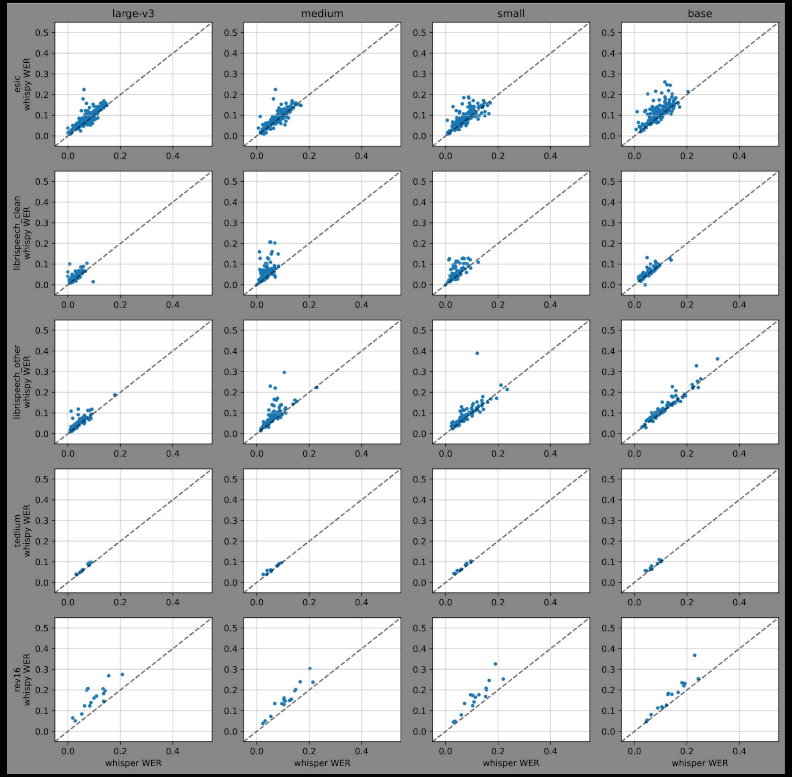
```
previous = 'Hello, this is a transcription for an audio that'    (len: 9)
current  = 'a transcription, for an audio that is recorded live' (len: 9)
distances(previous, current)
>>>
                                                'live' -> (index: 0, distance: 48)
                                      'recorded live' -> (index: 1, distance: 28)
                                   'is recorded live' -> (index: 2, distance: 19)
       MIN. DISTANCE ------> 'that is recorded live' -> (index: 3, distance: 16)
                         'audio that is recorded live' -> (index: 4, distance: 21)
                     'an audio that is recorded live' -> (index: 5, distance: 27)
                 'for an audio that is recorded live' -> (index: 6, distance: 30)
  'transcription, for an audio that is recorded live' -> (index: 7, distance: 33)
'a transcription, for an audio that is recorded live' -> (index: 8, distance: 47)

cutting_index = len(current.split(' ')) - argmin(distances(previous, current))
>>> 6

suggestion = ' '.join(current.split(' ')[cutting_index:])
>>> 'is recorded live'

previous + suggestion
>>> 'Hello, this is a transcription for an audio that is recorded live'
```

MAKE LOVE
NOT WER

OFFLINE COMPARISON



OFFLINE COMPARISON



TRANSCRIPTION DELAY

Table 1: Whispy processing times

dataset	model	trx	total
ESIC	large-v3	0.63 ± 0.07	0.88 ± 0.08
	medium	0.63 ± 0.07	0.90 ± 0.09
	small	0.28 ± 0.03	0.55 ± 0.06
	base	0.16 ± 0.02	0.44 ± 0.05
libri clean	large-v3	1.28 ± 0.14	1.56 ± 0.16
	medium	0.88 ± 0.08	1.03 ± 0.09
	small	0.33 ± 0.03	0.61 ± 0.07
	base	0.19 ± 0.02	0.49 ± 0.05
libri other	large-v3	1.16 ± 0.11	1.44 ± 0.16
	medium	0.74 ± 0.08	1.02 ± 0.11
	small	0.33 ± 0.03	0.61 ± 0.07
	base	0.18 ± 0.02	0.46 ± 0.06
tedlium	large-v3	1.24 ± 0.12	1.52 ± 0.14
	medium	0.73 ± 0.07	1.01 ± 0.09
	small	0.32 ± 0.03	0.59 ± 0.05
	base	0.18 ± 0.02	0.45 ± 0.04
rev16	large-v3	1.39 ± 0.11	1.66 ± 0.14
	medium	0.80 ± 0.06	1.06 ± 0.07
	small	0.35 ± 0.03	0.63 ± 0.05
	base	0.20 ± 0.02	0.47 ± 0.04

Average traversal time

```
large-v3 | 1.54

medium   | 1.03

small    | 0.61

base     | 0.46
```

WHERE ARE WE NOW?

Whispy: Adapting STT Whisper Models to Real-Time Environments

Antonio Bevilacqua[1], Paolo Saviano[1], Alessandro Amirante[1,2], and Simon Pietro Romano[1,2]

[1] Meetecho LTD, Napoli, Italy
https://www.meetecho.com/en
[2] University of Napoli Federico II
https://www.unina.it

Abstract. Large general-purpose transformer models have recently become the mainstay in the realm of speech analysis. In particular, Whisper

Fully operational system!

Hallucinations on steroids and resource management

Currently working on extensions (diarisation, summarisation...)

Paper under review for ECML-PKDD

BTW
I FOUND
THE CLICKER,
IT'S FINE

THANKS FOR LISTENING (AND READING)!

?? || /* */