

drea

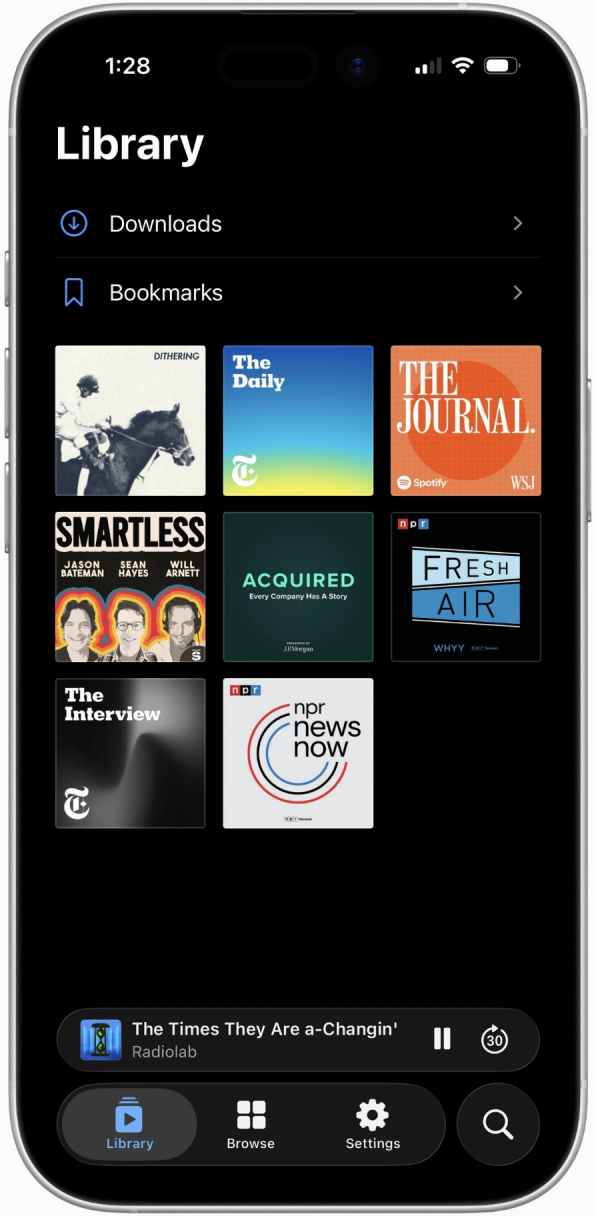
Efficiently detecting dynamic podcast ads via audio fingerprinting

Hamza Qayyum

Ad Blocker Developer Summit 2026 · Copenhagen, Denmark

drea

- Elegant, minimal podcast app
- Accurate ad detection and auto-skip
- Free
- iOS only, for now

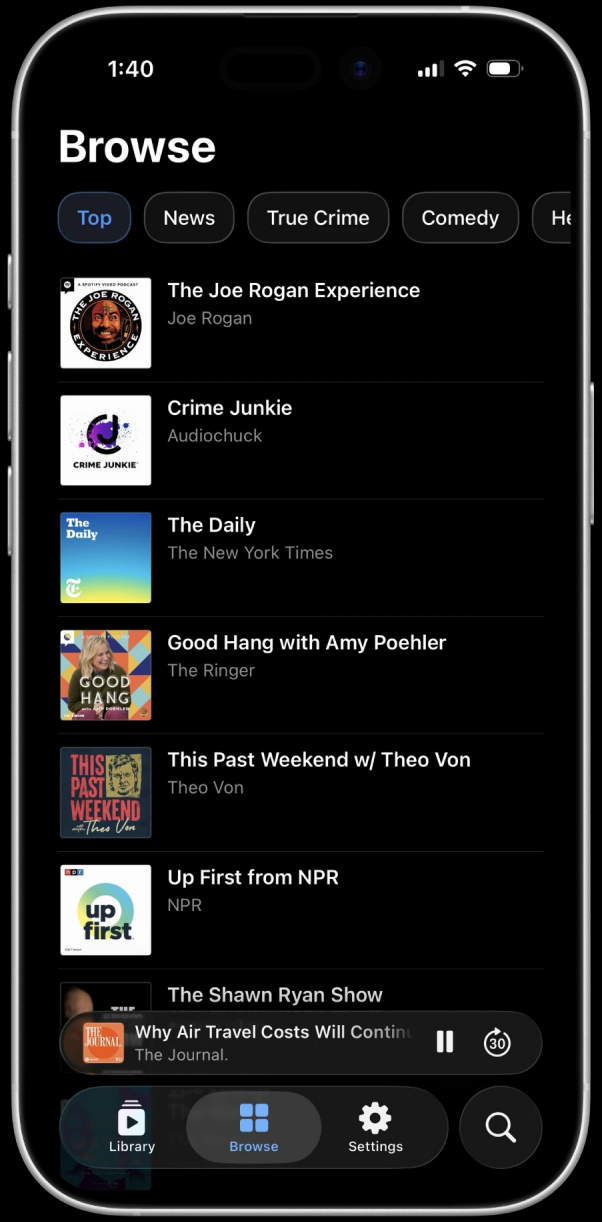


Motivation

No ad-skipping-capable podcast app I found matched the UI/UX quality of Apple Podcasts / Spotify, and none were free.


1:37
Ad finding
Finished
Found 3 ad segments
1
0:00 - 1:03
Length: 1m 3s
Play
2
13:18 - 15:31
Length: 2m 13s
Play
3
22:40 - 23:12
Length: 31s
Play
full studying the trees down here! So why bother?"
But then around the mid-1980s, a few ecologists around the world got curious and
Forests on Forests
Radiolab
Library
Browse
Settings


2:04
Bloomberg
Odd Lots
May 1
How Taiwan Became the World's Most Perilou
Odd Lots
11:41
-53:29
1x


1:40
Browse
Top
News
True Crime
Comedy
The Joe Rogan Experience
Joe Rogan
Crime Junkie
Audiochuck
The Daily
The New York Times
Good Hang with Amy Poehler
The Ringer
This Past Weekend w/ Theo Von
Theo Von
Up First from NPR
NPR
The Shawn Ryan Show
Why Air Travel Costs Will Contin
The Journal.
Library
Browse
Settings

drea

Past approaches to podcast ad detection

Adblock Radio (2018, Alexandre Storelli)

Neural classifier (~73k params) on spectral content + fingerprint bank of known ads

Identifying Advertisements in Podcasts (2025, Felix Wolf, Univ. of Hamburg master's thesis)

Neural classifier (~1M params) on audio + transcript sentence embeddings

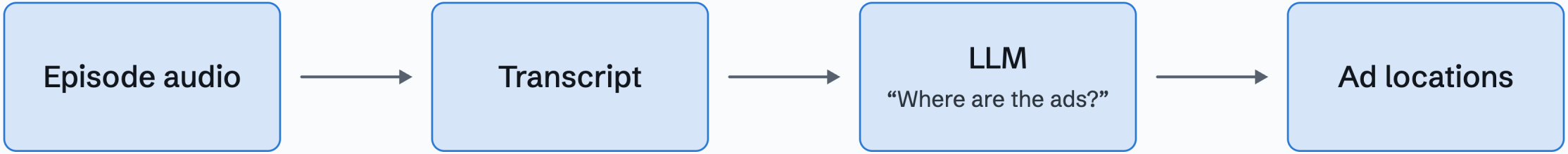
Both ran into accuracy issues.

Hypothesis: reliable podcast ad detection needs a lot of world knowledge, and thus a much higher parameter count.

LLMs (billions to trillions of params): a seemingly perfect solution

drea

A simple plan with LLMs



Done... right?

No. **Dynamic ads** throw a wrench into this.

drea

Two types of podcast ads

Static

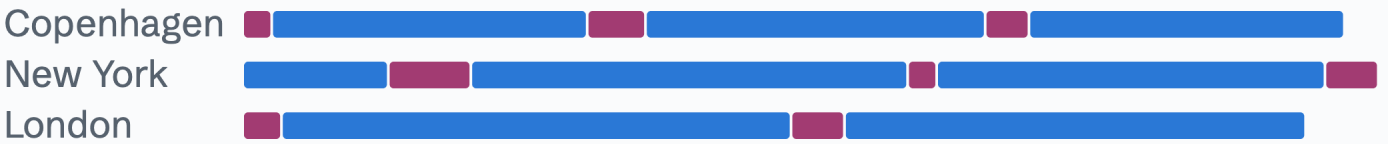
One version of the episode, so its ads never change.



✅ The simple plan works

Dynamic (programmatic) Popular since the mid-2010s

Ad placement and length vary with targeting: location, and possibly IP address, device OS, etc.

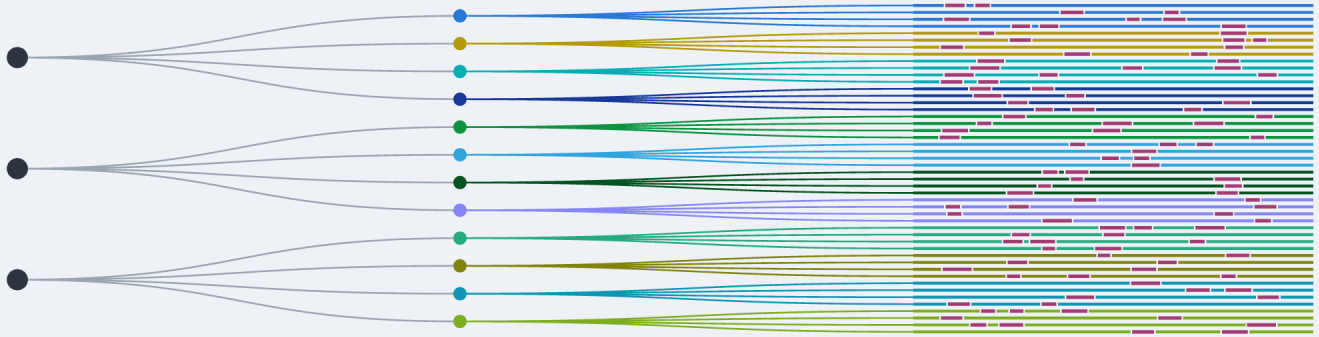


• • • and more, changing over time

Clean audio Ad

With dynamic ads, every version needs its own transcription and LLM call

Thousands of podcasts × **hundreds** of episodes × **tens to hundreds** of versions = **millions** of LLM calls



High latency

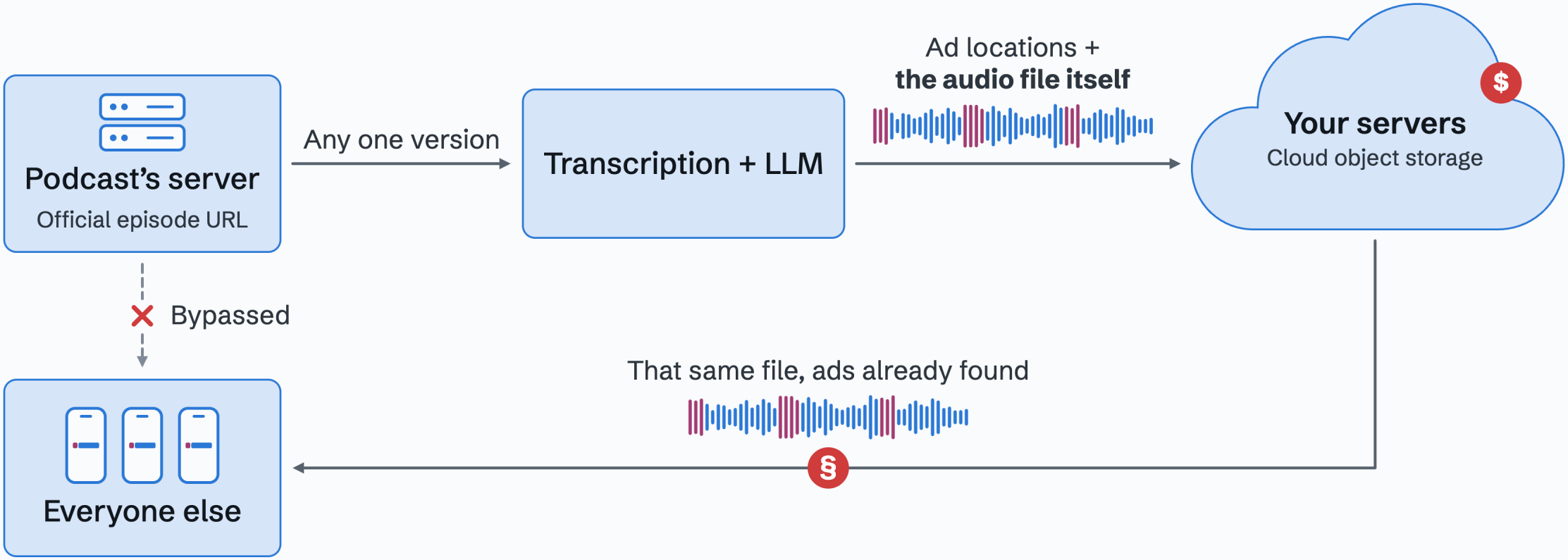
Wait for transcription and LLM calls

High cost

LLM API bill will bankrupt you and your future generations

drea

The naive solution: rehost the audio



§ Illegal

Rehosting and redistributing copyrighted audio without a license. It must always come from the official URL.

$ Costly

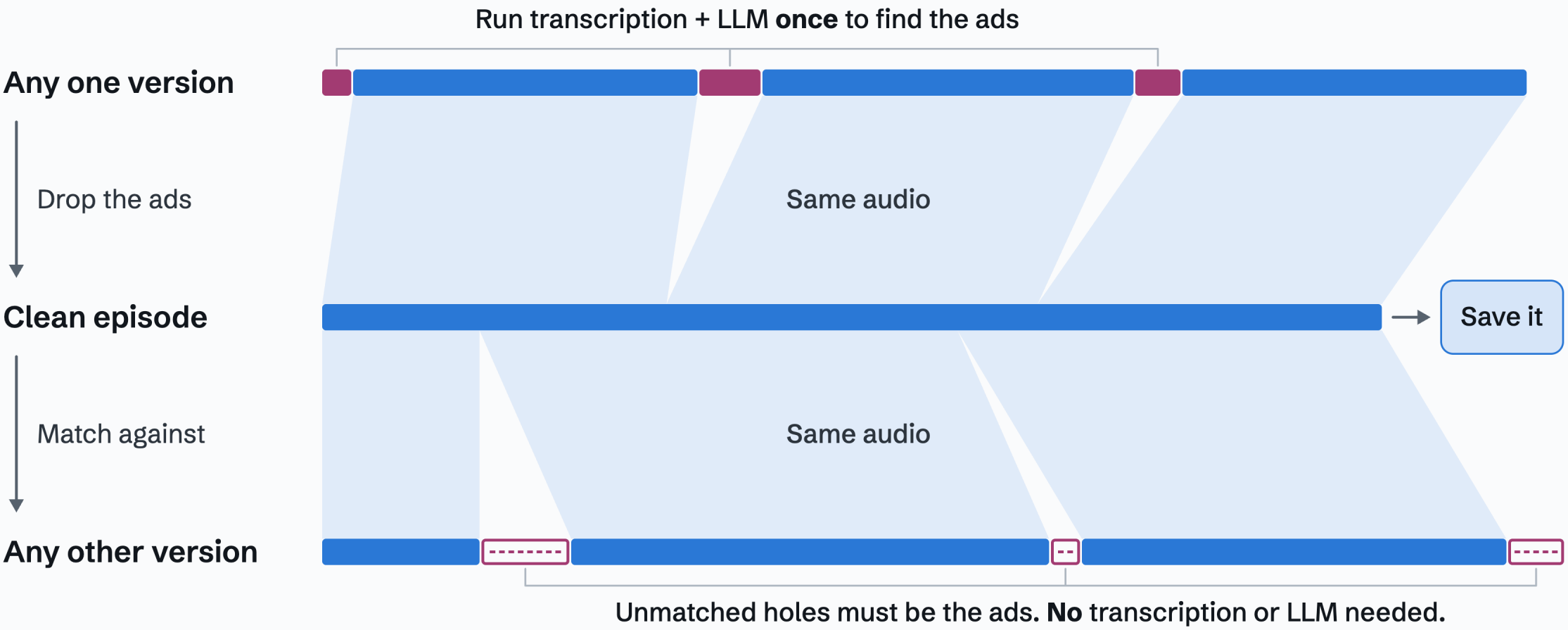
Every episode stored in cloud object storage and streamed from there. Three hours at 128 kbps is about 170 MB.

drea

What stays the same across versions

Ads and their locations change

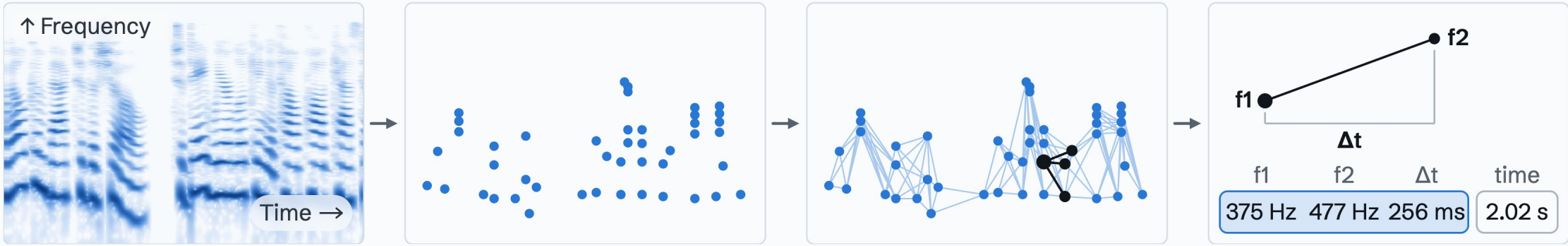
The clean audio never changes



drea

Shazam-style fingerprints

How we save the clean episode; invented by Avery Wang at Shazam (founded 1999).



1. Spectrogram

Slice the audio into short windows and measure how loud each frequency is in each one.

2. Peaks

Keep only points that are the loudest within ±63 Hz and ±64 ms, up to 15 per second.

3. Pairs

Pair each peak with 3 peaks after it.

4. Hashes

Each pair becomes a 32-bit hash of its two frequencies and their time gap, saved with its timestamp.

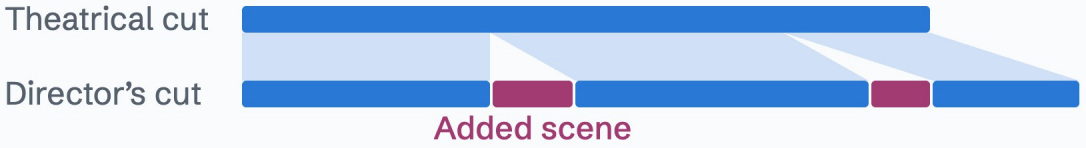
- **Cannot be turned back into audio**
  Only a few frequencies and times per second survive, with no loudness or phase.
- **Long-standing legal precedent**
  Considered *facts about the work*, not the work itself; safe to store (and redistribute)
- **Tiny: about 0.4 MB per hour**
  Alternatives like MFCCs are much heavier.

Shazam fingerprints were invented for ***identification***, *e.g. identifying a song*.
We need to do ***accurate matching***. Can they be used for that?

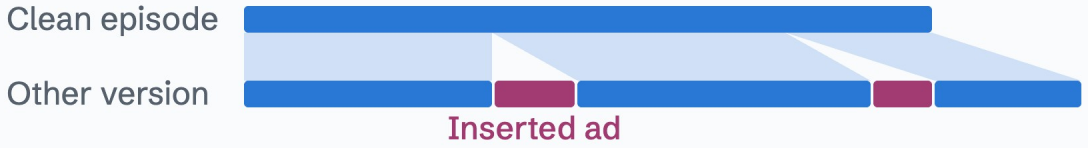
Movie synchronization by audio landmark matching

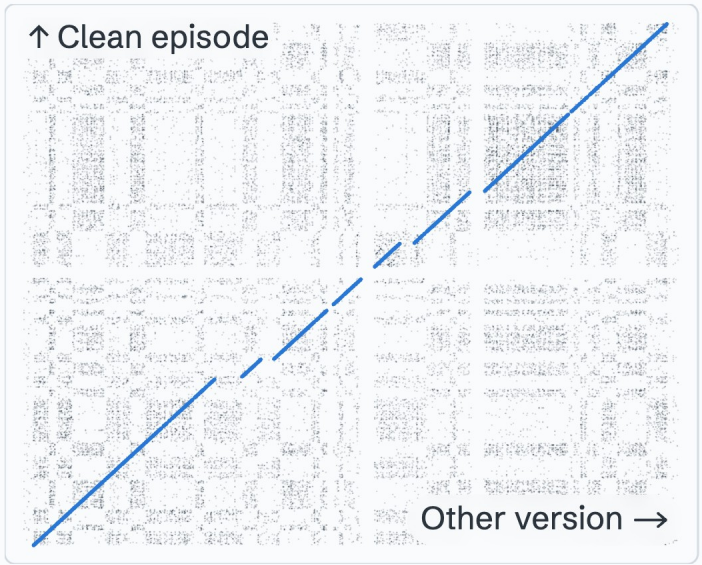
Ngoc Q. K. Duong and Franck Thudor at Technicolor (2013, IEEE ICASSP)

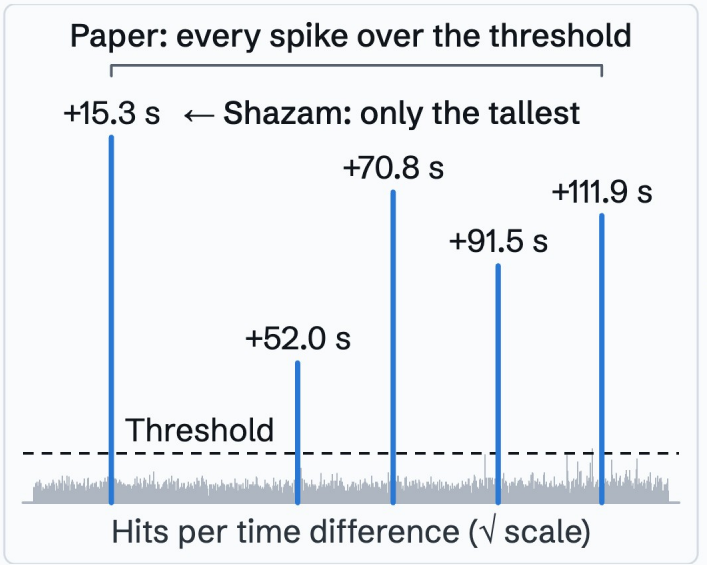
Paper: one film, many cuts

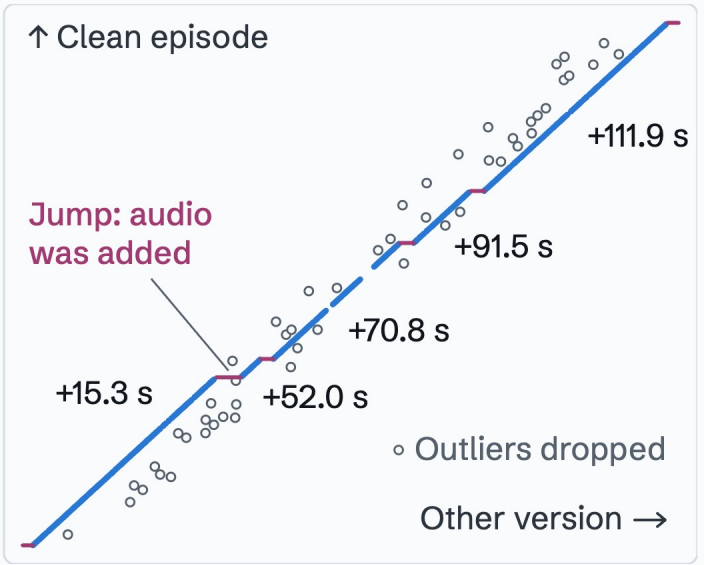


drea: one episode, many versions









1. Look up every hash

A shared hash is a hit: a pair of times. Of 53,719 hits here, same audio falls on diagonals; accidents scatter.

2. Count the time differences

Time difference = version time − clean time. All hits on a diagonal share one, so each diagonal piles up into one spike.

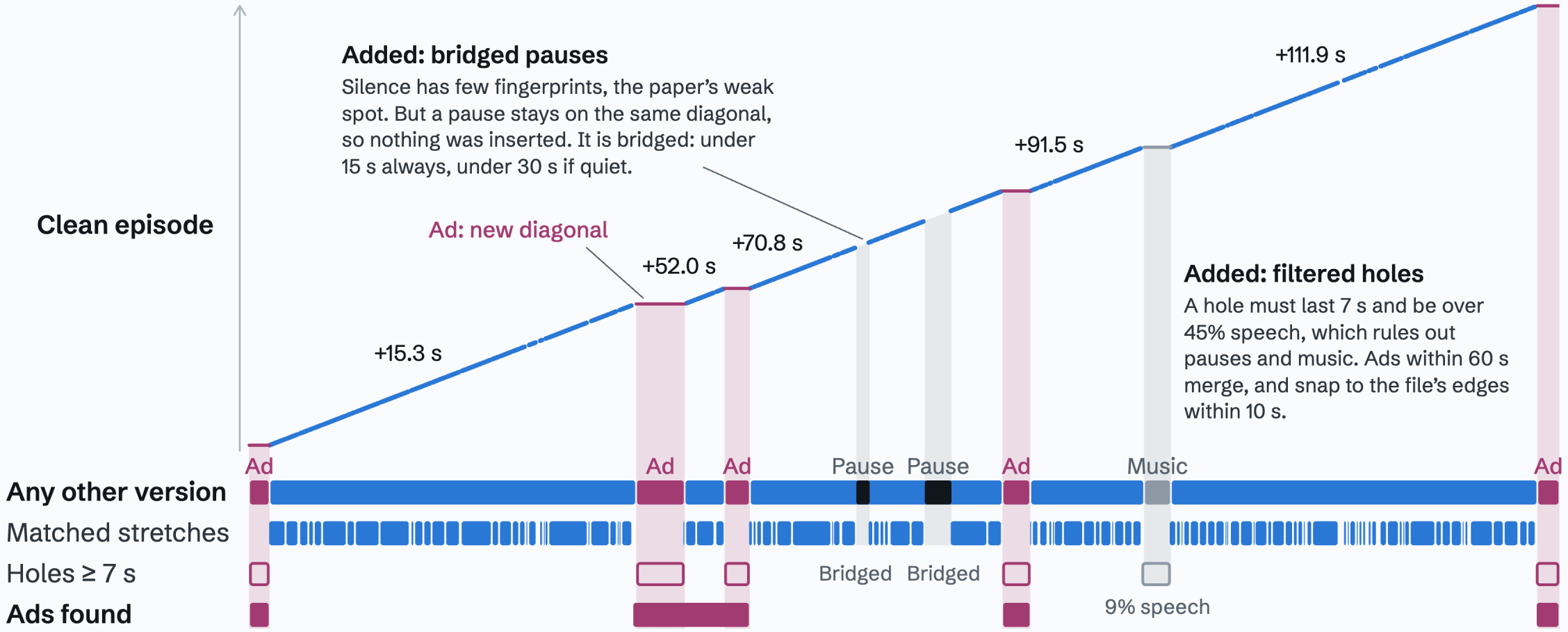
3. Cluster each diagonal

Keep hits near a spike (to remove accidental hits), cluster them and drop small clusters. Each stretch now has an exact start and end.

drea

Finding ads by matching clean audio fingerprints

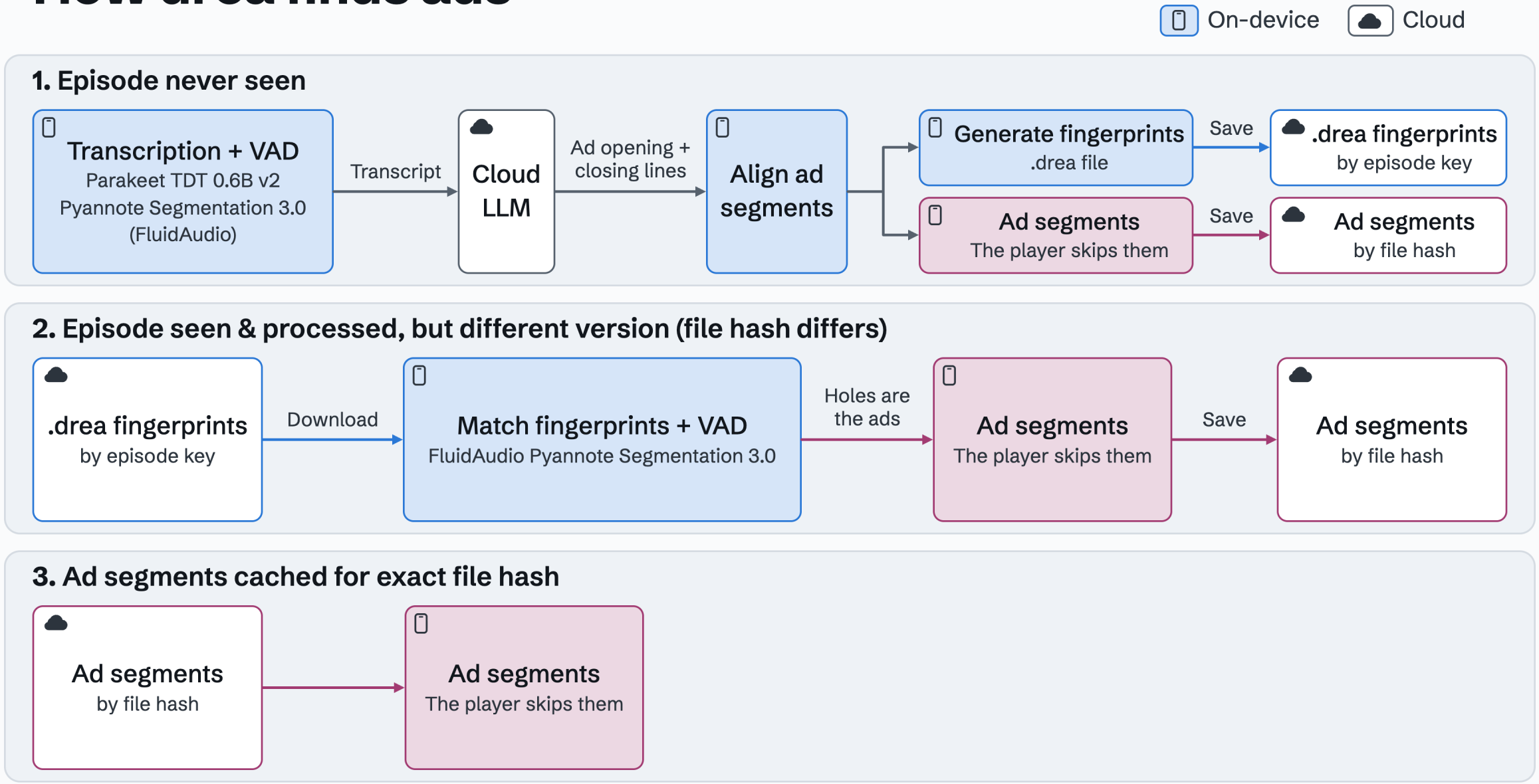
Allows us to only have to transcribe + call LLM once per episode, ever.

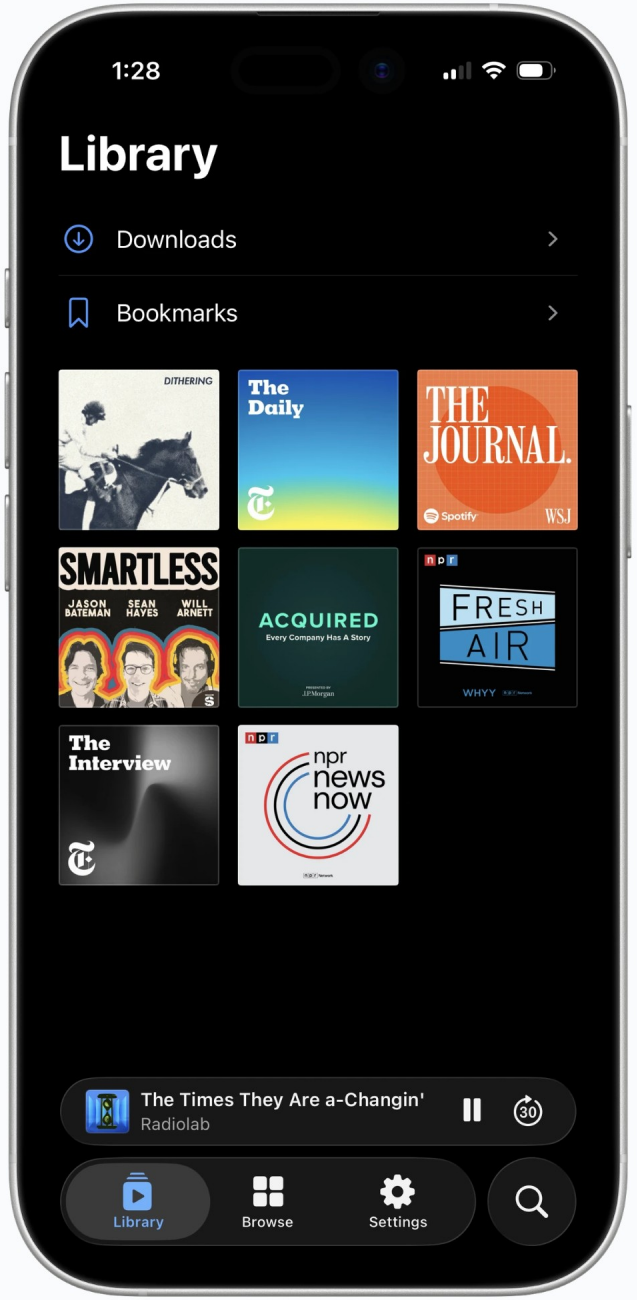


Real run on a synthetic 15-minute test episode, using a basic TTS model

drea

How drea finds ads





drea

Download on the App Store