Finding what you know and Finding out what you don't from Audio

Sue Johnson

February 21st, 2000

Cambridge University Engineering Department

Outline

Searching the Audio

- Background
  1. Why search the audio?
  2. Why is it difficult?
- Details
  1. What options do we have?
  2. Learning from speech recognition
  3. How do we do it?
- Results
  1. Searching broadcast news audio
  2. How fast and small can we go?

Application within Spoken Document Retrieval (SDR)

1. Story-Known SDR
2. Story-Unknown SDR
3. Advert detection
4. Using Broadcast Structure

Why Search Audio?

- Huge, constantly increasing amount of audio data available
- Very time consuming to listen to it
- Allows match of non-speech events (music, laughter, coughing)
- Allows match of particular speaker, intonation etc.
- Allows given instance to be found e.g. "I did not have sexual relations with that woman"
- ASR is not 100% accurate
- High quality ASR is generally "slow" (e.g. x10RT)
- ASR is not always possible (e.g. on small devices)

Why is it difficult?

- Text-based Searching

  1. Small, discrete set of characters
  2. Character set is unambiguous and well-defined
  3. Perception of a character requires only 1 character
  4. Simple "is-equal-to" operator can be used
  5. Dimensionality is small (e.g. 1 word ~ 5 chars)

- Audio-based Searching

  1. Continuous audio stream - infinite possibilities
  2. Must be digitized/coded (brings in ambiguity)
  3. Must be sampled (brings in asynchrony)
  4. Perception of a "sound" depends on a sequence of samples
  5. "fuzzy matching" must be used
  6. Dimensionality is huge (e.g. 1 word ~ 16kHz x 0.4s)

What options do we have?

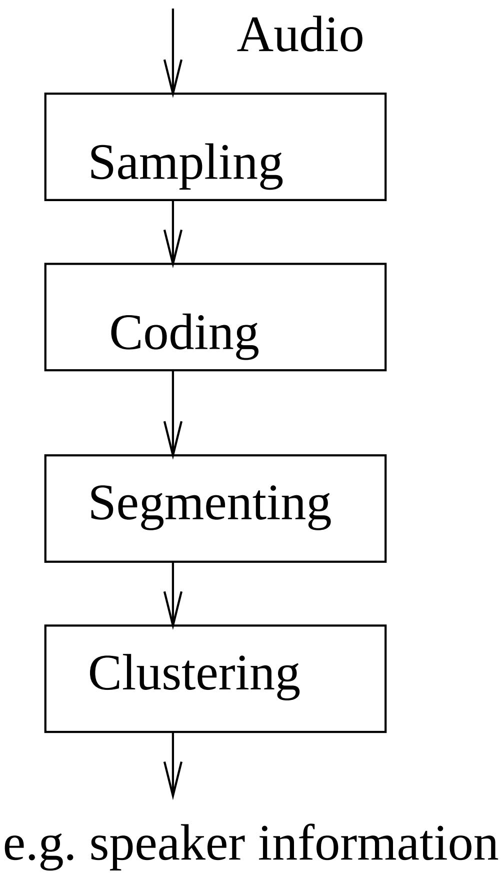
Content-based IR techniques to find "similar" things
e.g. `http://www.musclefish.com/cbrdemo.html`

Exact-matches

- Time-series active search
  - e.g. search amplitude directly in time domain
  - speed up by skipping when no close match (see Gavin!!)
- Frame-based Methods
  - Speed up search at expense of synchrony (100Hz)
  - Use "more relevant" params eg. pitch, ZCR, Energy
  - build a vector for each frame
  - match sequence of vectors with e.g. cosine distance
- Block-based Methods
  - Speed up further at expense of synchrony (e.g. 1Hz)
  - Use a higher order statistics (e.g. covariance)
  - Capture generic properties but lose sequence info

Learning from Speech Recognition

Speech Recogniser front end



- Sampling discretizes the signal
- Coding introduces more perceptual parameters
- Clustering and segmentation look for similar properties

How do we do it

We borrow the representation from speaker clustering since
1. The task is similar (and shown to work well)
2. The coded data is available to us at no extra cost

The method

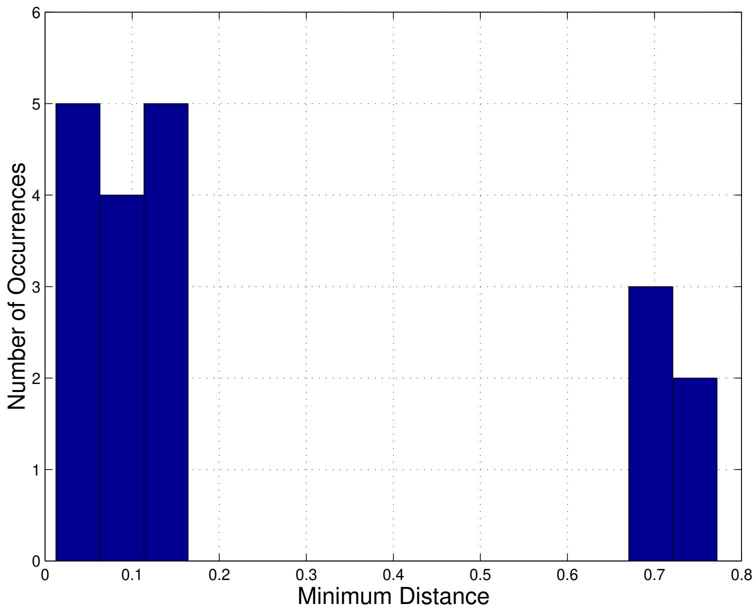
- Take the sampled/coded data from ASR (MFCCs)
- Define a segment of speech by the required length of audio
- Represent the segment by its covariance matrix Σ
- Find the AHS distance between the matrices:

$$d(X,Y) = \log[tr(\Sigma_y \Sigma_x^{-1}) \cdot tr(\Sigma_x \Sigma_y^{-1})] - 2\log(D)$$

- Compare to a small threshold
- Move the segment window on by ϵ and repeat

Searching Broadcast News Audio

- End-Point Detection
- 6.63s piece of *cue-audio* for ABC News
- Define windows 6.63s @ 0.5s shift
- Find minimum distance in last 5 minutes



- Clearly separated (sep. factor = 4.53)
- Potential for speed up
  (e.g. compare to threshold, increase window shift)

How fast can we go?

- Reducing the number of params (static only, diag Σ)

Features Used	Dist.	Cov.	Speed (xRT)	RMS Error(s)	Sep. Ratio
+deriv	AHS	Full	0.0256	0.189	4.53
+deriv	Euc.	Diag.	0.0083	0.148	1.50
static	AHS	Full	0.0053	0.173	3.01
static	Euc.	Diag.	0.0030	0.148	1.50

AHS, static only, full Σ, @0.5s = 200FxRT

- Increasing the Window Shift

Window Shift	AHS +deriv	AHS static	Euclid static
0.5s	4.53	3.01	1.50
1.0s	4.06	2.45	1.23
2.0s	2.36	1.73	1.01
3.0s	2.18	1.56	–
4.0s	2.27	1.60	–
5.0s	1.87	1.17	–

AHS, static only, full Σ, @5.0s = 2000FxRT

How small can we go?

- Reducing the size of Cue-Audio

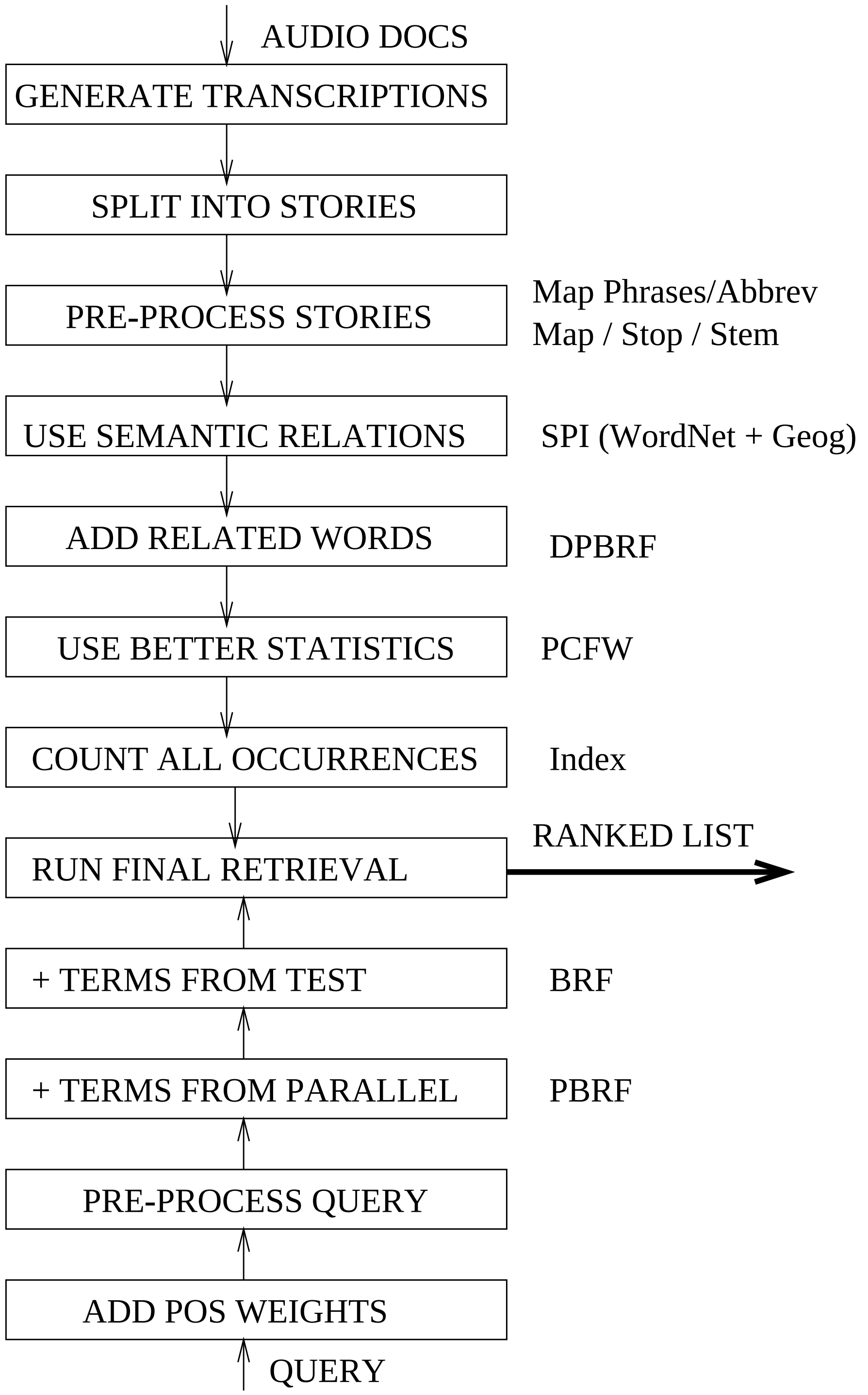
Size (s)	Shift (s)	Misses	Speed (xRT)	Sep. Ratio
6.0	0.5	0	0.0050	3.45
4.0	0.5	0	0.0040	2.62
2.0	0.5	0	0.0030	2.26
1.0	0.5	0	0.0026	1.53
0.5	0.25	3	0.0023	1.02
0.5	0.1	0	0.0041	2.18
0.3	0.1	0	0.0037	1.51
0.2	0.1	1	0.0034	1.07
0.2	0.05	1	0.0050	1.06

- 1s is sufficient (when shift = 0.5s)
- 0.3s is sufficient (when shift = 0.1s)
- would need more cue-audio for text search

Spoken Document Retrieval

- SDR is trying to retrieve information from documents which originated as speech.
- Evaluated within TREC on broadcast news
- TREC-8 had 500 hours of audio with 902 broadcasts from ABC, CNN, PRI, VOA spanning February 1st to June 30th 1998
- 50 text queries/relevance assessments given by NIST
- Recognition must be done *on-line*
- Retrieval is done *retrospectively*
- Recognition & retrieval must be completely automatic
- Most people run an ASR system, then a word-based IR system on the transcriptions, but other approaches are possible (e.g. phoneme based)
- Two tasks -
  - Story known (SK), manually-generated story boundaries provided
  - Story unknown (SU), no manual knowledge used.

Our SK-SDR system



General SU-SDR Systems

Segmentation-based methods

1. Run a "story-segmentation" algorithm
2. Run Retreiver as for SK case

+ makes query-independent stories
+ makes process much faster (no overlap, no recombination)
– results so far proved to be poor

Windowing/Recombination-based methods

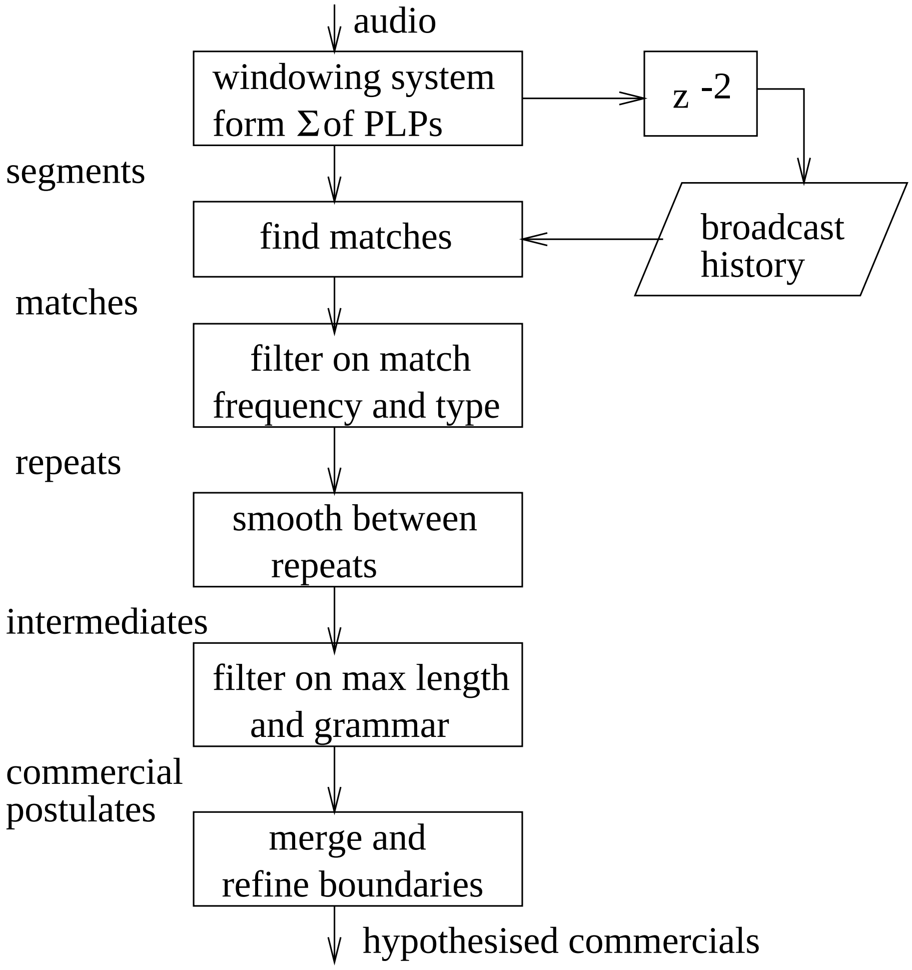
1. Apply sliding window system
2. Run Retriever (with no DPBRF)
3. Merge nearby/overlapping relevant stories

+ results proved good
– no "stories" per se produced
– very slow to process
  (20 minutes at query time for TREC-8)

Removing Adverts

Adverts:

- Are irrelevant to general queries
- Provide indicatations of structure within the broadcast
- Slow down recognition and retrieval
- Generally get re-broadcast several times



Removing Adverts - Results

1. TREC-7 ABC data (8.7 hrs, retrospective)

T_s (s)	Non-story Rejection	Story Rejection
0 (none)	59.41%	0.17%
60	70.90%	0.17%
90	73.34%	0.45%

2. TREC-8 data (500 hours, on-line)

Broad.	Non-Stories	Stories	Total
CNN	26.19hr=35.7%	2822s=0.46%	27.0hrs=11.0%
ABC	12.78hr=65.5%	28s=0.02%	12.8hrs=20.5%
PRI	1.93hr=16.6%	297s=0.10%	2.0hrs= 2.2%
VOA	0.47hr= 5.0%	132s=0.04%	0.5hrs= 0.5%
ALL	**41.4hrs=36.3%**	**0.9hrs=0.2%**	**42.3hrs=8.4%**

- 8.4% (42 hours=26 CPU-days) of data thrown away
- 97.8% labeled "non-story" by assessors
- Average Precision increased by 1.2% relative
- A system removing 63hrs data was 95.0% non-story

Using Broadcast Structure

Step	Description
GENERATE TRANSCRIPTIONS	eliminate commercials eliminate music and silence
GENERATE SLIDING WINDOWS	generate 30s windows @ 15s enforce boundaries at gaps >5s truncate shorter windows remove short/few-word windows
RUN RETRIEVER	as for SK system with no document expansion
POST-PROCESSING	enforce boundaries at gaps > 5s take only first for windows within 75s output top 1000 per query

- Find repeats (adverts, jingles etc)
- Force boundaries at these breaks
- Allows merge-time in post-processing to be increased
- Increases precision

Summary

Finding what you know in Audio

- Fast, accurate, robust audio search method described
- Methods for speeding up search discussed
- Results for broadcast news application given

Finding out what you don't know from Audio

- Story-Known SDR system described
- Methods of running a story-unknown system given
- Automatic advert removal speeds up recognition and helps retrieval
- Associated broadcast structure can also be used

For more info

http://svr-www.eng.cam.ac.uk/~sej28/papers/
http://svr-www.eng.cam.ac.uk/Research/Projects/mdr