Automatic Speech Recognition

ASR Tasks and Methods

dr inż. Jakub Gałka, AGH

Types of ASR

- Isolated word (phrase) recognition   (“stop”, “one”)
- Concatenated words recognition (“zero zero five”, “start playback”)
- Phrase recognition
  - Grammar rules, limited dictionary, SRGS, ABNF (“pay *two hundred* to *John*”)
  - Free grammar, large dictionary, Language Grammar: e.g. N-Gram (“What kind of party is it”)
- LVCSR - Dictation
  - Large vocabulary: 100 000-300 000 words, or more (“Once upon a time…”)
  - Often domain-specific language models
    (e.g. medical: “He has had hypertension and hyperlipidemia since 1990”)
  - Lot of different web-APIs (Google Speech API)
- Wake-word / Hot-word detection (“OK Google!”, “Alexa!”)
- Word-spotting, word / phrase search

ASR Performance

- Computational performance
  - latency,   < 500ms, <1000ms
  - computational scalability: how many audio streams in *real-time* on a CPU core or GPGPU (FLOPS)
  - memory usage (RAM)
- Accuracy (%)
  - Word/Phrase Recognition Rate (%), Phone Error Rate (%)
  - **Word Error Rate** (%), Dictation: WERR < 2%
  - False Positive Rate, False Negative Rate (%) ⇔ False Alarm Rate, Missed Detection Rate
    - e.g False Alarm per Hour (e.g. for some specific noise dB level)
  - Precision, Recall, f-score, ROC, AUC (general detection performance)
- Recognition Confidence: e.g. (0 - 100 %)
  - confidence threshold, e.g. MAP Probability
- Robustness: how noise (e.g. SNR) impacts performance
  - noise: cocktail party, babble noise, street noise, office noise, speech codec noise
  - channel: mic type, transmission channel, sound equalization, reverberation
  - speaker variation

Word Error Rate (%)

$$\text{Word Error Rate} \;=\; 100 \times \frac{\text{Insertions} + \text{Substitutions} + \text{Deletions}}{\text{Total Words in Correct Transcript}}$$

REF:	i	***	**	UM	the	PHONE	IS	i	LEFT	THE	portable	****	PHONE	UPSTAIRS	last	night
HYP:	i	GOT	IT	TO	the	*****	FULLEST	i	LOVE	TO	portable	FORM	OF	STORES	last	night
Eval:		I	I	S		D	S		S	S		I	S	S		

$$\text{Word Error Rate} \;=\; 100\frac{6+3+1}{13} = 76.9\%$$

ASR - Automatic Speech recognition problem statement

https://web.stanford.edu/~jurafsky/slp3/

Convert speech audio to text or tokens (not much of NLU)

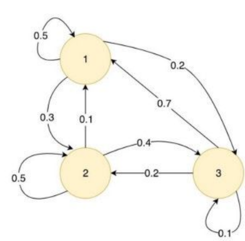
Which word / character sequence is most likely given the acoustical observations sequence?

$$p(y_1,\ldots,y_n) = \prod_{i=1}^{n} p(y_i|y_1,\ldots,y_{i-1},X)$$

$$\hat{y}_i \;=\; \text{argmax}_{\text{char}\in\,\text{Alphabet}} P(\text{char}|y_1 \ldots y_{i-1},X)$$

HMM - Hidden Markov Model

$$\mathrm{P}(X_{n+1}=x \mid \underbrace{X_1=x_1, X_2=x_2, \ldots,}_{\text{can be ignored}} X_n=x_n) = \mathrm{P}(X_{n+1}=x \mid X_n=x_n)$$

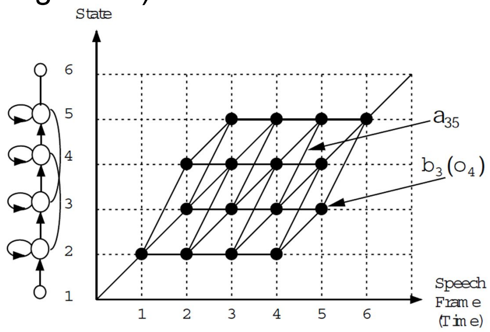


Markov process

$$\begin{array}{cc} & \begin{array}{ccc} s_1 & s_2 & s_3 \end{array} \\ \begin{array}{c} s_1 \\ s_2 \\ s_3 \end{array} & \begin{bmatrix} 0.5 & 0.1 & 0.7 \\ 0.3 & 0.5 & 0.2 \\ 0.2 & 0.4 & 0.1 \end{bmatrix} \end{array}$$

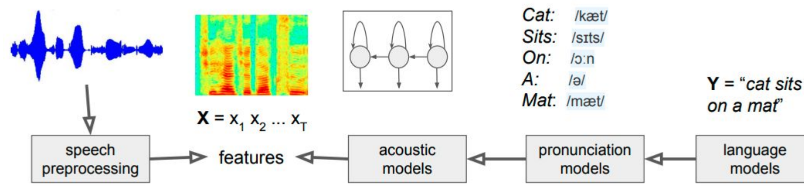
Transition matrix

Hidden Markov Model - best state sequence (Viterbi algorithm)



HMM speech modeling

source: https://web.stanford.edu/class/archive/cs/cs224n/cs224n.1174/lectures/cs224n-2017-lecture12.pdf

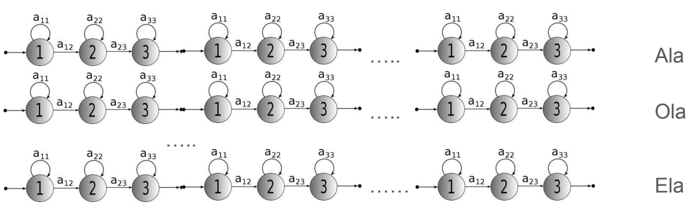


$$Y^* = \arg\max_{Y} p(\mathbf{X}|\mathbf{Y})\, p(\mathbf{Y})$$

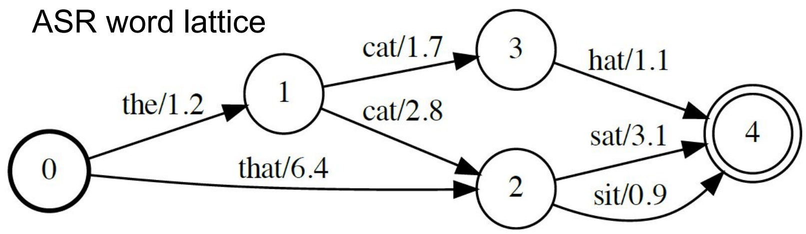
HMM recognition from hypotheses list

GMM - acoustic model likelihood (observation likelihoods)
HMM - phonetic phrase model (e.g. triphone) (underlying phonetic state model)

By concatenating several HMMs for each word or phrase, we can calculate the probability of the words given acoustic data and pick the most likely word, or word sequence.



ASR word lattice

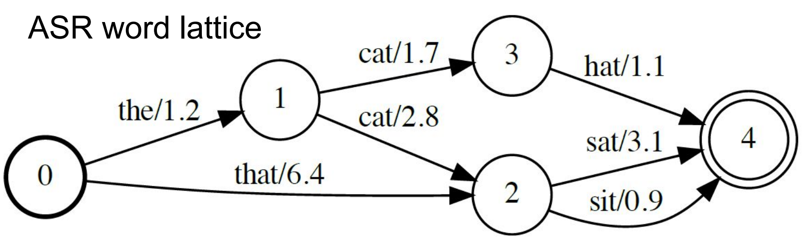


N-Best list:
"that sat": 9.5
"that sit": 7.3
"the cat sat": 7.0
"the cat sit": 4.9
"the cat hat": 4.0

=== Language Model rescoring ⇒

"that sit": 12.0
"the cat sat": 10.2
"the cat sit": 7.0
"that sat": 5.4
"the cat hat": 2.8

ASR word lattice



$$score(Y|X) = \frac{1}{|Y|_c} \log P(Y|X) + \lambda \log P_{LM}(Y)$$

N-Best list:
"that sat": 9.5
"that sit": 7.3
"the cat sat": 7.0
"the cat sit": 4.9
"the cat hat": 4.0

=== Language Model rescoring ⇒

"that sit": 12.0
"the cat sat": 10.2
"the cat sit": 7.0
"that sat": 5.4
"the cat hat": 2.8

Combining Acoustic and Language Models

$$P(W|X) = p(X|W) * P(W) / P(X)$$

$$\log P(W|X) = \log p(X|W) + \log P(W) - \log P(X)$$

$$score(Y|X) = \frac{1}{|Y|_c} \log P(Y|X) + \lambda \log P_{LM}(Y)$$

N-Best list:
"that sat": 9.5
"that sit": 7.3
"the cat sat": 7.0
"the cat sit": 4.9
"the cat hat": 4.0

=== Language Model rescoring ⇒

"that sit": 12.0
"the cat sat": 10.2
"the cat sit": 7.0
"that sat": 5.4
"the cat hat": 2.8

Language models

Rule-based (FST, grammars, SRGS, BNF)

Statistical (N-gram, bag-of-words)

Deep Learning

- Word2vec (skip-gram), FastText (from FB (Meta))
- BERT - Bidirectional Encoder Representations from Transformers
- GPT-2, GPT-3 (Generative Pre-trained Transformer)

Rule-Based Language modeling (SRGS Example)

Chomsky language classification

- 0 Recursively enumerable grammars –recognizable by a Turing machine
- 1 Context-sensitive grammars –recognizable by the linear bounded automaton
- 2 Context-free grammars - recognizable by the pushdown automaton
- 3 Regular grammars –recognizable by the finite state automaton

```
#BNF+EM V2.1;
!grammar ASRENG-US;
!start <main>;
!pronounce “capricciosa” PRONAS “caprikiosa” | PRONAS “caprichioza”;
<main>: <order> | <time_left>;
<time_left>: How (much | many) time (left | remaining) [for (our | my) order];
<order>: <verb> !repeat((a | <number>) (<pizza> | <drinks>) [(and | with) [<verb>]], 1, *);
<verb>: I (want | would like);
<number>: !tag(NUMBER, 1 | 2 | 3);
<pizza>: [pizza] !tag(PIZZA_TYPE, margherita | prosciutto e funghi | capricciosa | vegetariana | calzone);
<drinks>: [!tag(DRINK_FORMAT, glass | bottle) of] !tag(DRINK_TYPE, water | coca | wine | beer);
```

“I want a capricciosa and a bottle of water.”

N-Gram language modeling

P(X1...Xn) = P(X1)P(X2|X1)P(X3|X1:2)...P(Xn|X1:n−1)

Uni-gram

P(w1w2w3w4) = P(w1)P(w2)P(w3)P(w4)

Bi-gram

P(w1w2w3w4) = P(W4|W3)P(W3|W2)P(W2|W1)P(W1)

Tri-gram

P(w1w2w3) = P(W4|W2W3)P(W3|W1W2)P(W2|W1)P(W1)

Bi-gram model example
use model *smoothing* to get rid of zeros

	i	want	to	eat	chinese	food	lunch	spend
i	5	827	0	9	0	0	0	2
want	2	0	608	1	6	6	5	1
to	2	0	4	686	2	0	6	211
eat	0	0	2	0	16	2	42	0
chinese	1	0	0	0	0	82	1	0
food	15	0	15	0	1	4	0	0
lunch	2	0	0	0	0	1	0	0
spend	1	0	1	0	0	0	0	0

Figure 3.1 Bigram counts for eight of the words (out of $V = 1446$) in the Berkeley Restaurant Project corpus of 9332 sentences. Zero counts are in gray.

	i	want	to	eat	chinese	food	lunch	spend
i	0.002	0.33	0	0.0036	0	0	0	0.00079
want	0.0022	0	0.66	0.0011	0.0065	0.0065	0.0054	0.0011
to	0.00083	0	0.0017	0.28	0.00083	0	0.0025	0.087
eat	0	0	0.0027	0	0.021	0.0027	0.056	0
chinese	0.0063	0	0	0	0	0.52	0.0063	0
food	0.014	0	0.014	0	0.00092	0.0037	0	0
lunch	0.0059	0	0	0	0	0.0029	0	0
spend	0.0036	0	0.0036	0	0	0	0	0

Figure 3.2 Bigram probabilities for eight words in the Berkeley Restaurant Project corpus of 9332 sentences. Zero probabilities are in gray.

P(<s> i want english food </s>)
= P(i|<s>)P(want|i)P(english|want)
P(food|english)P(</s>|food)
= .25×.33×.0011×0.5×0.68
= .000031

hint: *use log_prob for calculations*

NN Language Models: e.g. Word2Vec model

- Can be used to predict word likelihood given the preceeding words
- FastText from Meta (Facebook) library
- Word2Vec training:

Thou shalt not make $\boxed{\text{a machine in}}$ the likeness of a human mind

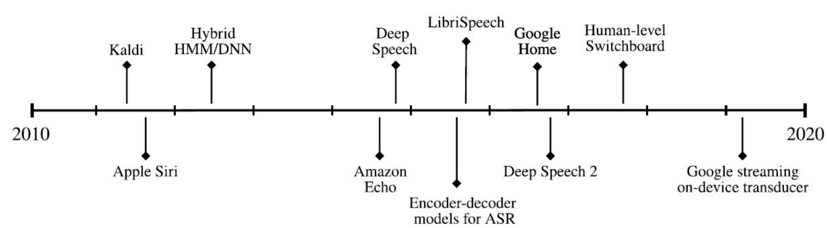
Sliding window across running text

thou	shalt	not	make	a	machine	in	the	...
thou	shalt	not	make	a	machine	in	the	
thou	shalt	not	make	a	machine	in	the	
thou	shalt	not	make	a	machine	in	the	
thou	shalt	not	make	a	machine	in	the	

Dataset

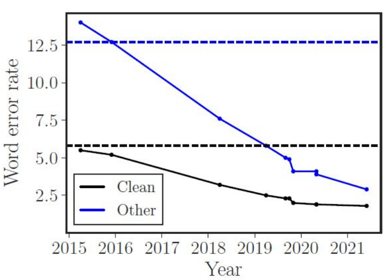
input 1	input 2	output
thou	shalt	not
shalt	not	make
not	make	a
make	a	machine
a	machine	in

Deep Architectures for ASR

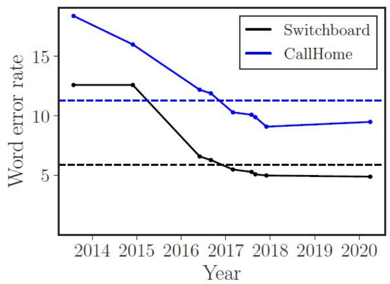


Good resource for modern ASRs learning: https://web.stanford.edu/~jurafsky/slp3/

ASR Accuracy



(a) LibriSpeech

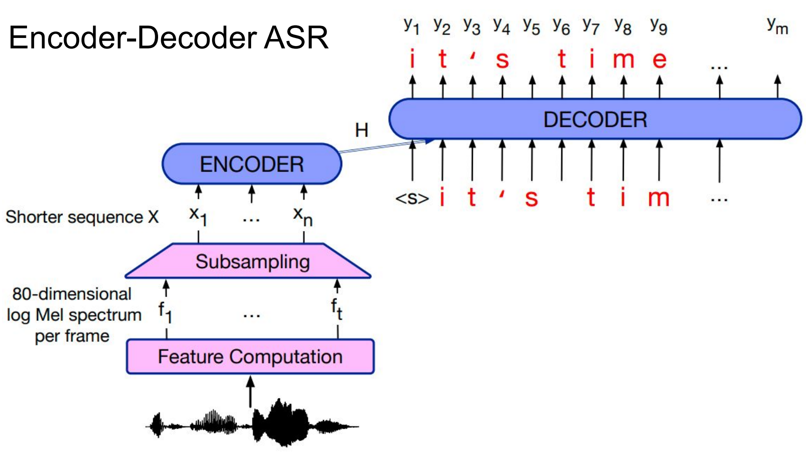


(b) Switchboard Hub5'00

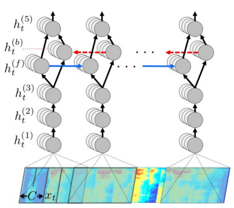
ASR Benchmarks (2020)

English Tasks	**WER%**
LibriSpeech audiobooks 960hour clean	1.4
LibriSpeech audiobooks 960hour other	2.6
Switchboard telephone conversations between strangers	5.8
CALLHOME telephone conversations between family	11.0
Sociolinguistic interviews, CORAAL (AAL)	27.0
CHiMe5 dinner parties with body-worn microphones	47.9
CHiMe5 dinner parties with distant microphones	81.3
Chinese (Mandarin) Tasks	**CER%**
AISHELL-1 Mandarin read speech corpus	6.7
HKUST Mandarin Chinese telephone conversations	23.5

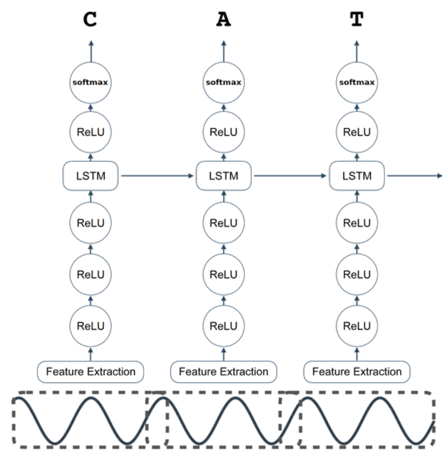
Encoder-Decoder ASR



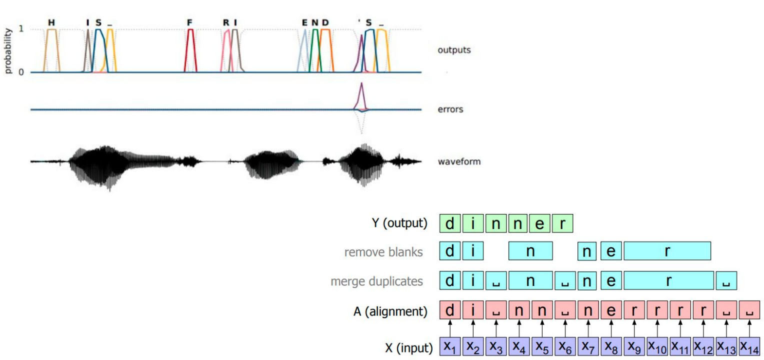
Deep Speech (2014)



System	Clean (94)	Noisy (82)	Combined (176)
Apple Dictation	14.24	43.76	26.73
Bing Speech	11.73	36.12	22.05
Google API	6.64	30.47	16.72
wit.ai	7.94	35.06	19.41
Deep Speech	**6.56**	**19.06**	**11.85**



CTC - Connectionist Temporal Classification



SOTA models and methods

https://web.stanford.edu/class/archive/cs/cs224n/cs224n.1174/lectures/cs224n-2017-lecture12.pdf

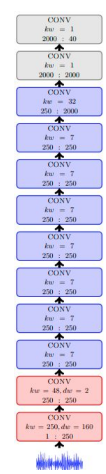
- Back-end ASR with DNN-HMM LM
- end-to-end, effective with huge datasets
  - Transformer with conv layers (Conformer) and CTC CTC (Connectionist Temporal Classification) / LAS (Listen Attend and Spell)
  - Attention models (ResNet + self-attention) and LM rescoring.

- Some ASR Frameworks

wav2letter++ (https://arxiv.org/abs/1812.07625)

Deep Speech (https://github.com/mozilla/DeepSpeech)

Kaldi (http://kaldi-asr.org/)



ASR Datasets

Speech + transcription

- no transcription
- phone-level transcription
- word-level transcription
- phrase-level transcription
- no alignment (for pre-training or forced-alignment required)

https://www.openslr.org/resources.php

https://commonvoice.mozilla.org/

ASR API

REST, RPC (Google API: https://cloud.google.com/speech-to-text/docs/apis)

API requirements

- synchronous, asynchronous
- confidence measure,
- N-best list with scorings (for rescoring)

API methods (e.g. *createRecognizerInstance, getRecognition, setQuality, setLM, getAlternatives, setLanguageModel*, …)

Some APIs: https://medium.com/sciforce/automatic-speech-recognition-asr-systems-compared-6ad5e54fd65f

VoiceXML,

https://www.w3.org/Voice/Guide/

```
<?xml version="1.0"?>
<vxml version="2.0">
<menu>
  <prompt>
    Say one of: <enumerate/>
  </prompt>
  <choice next="http://www.sports.example/start.vxml">
     Sports
  </choice>
  <choice next="http://www.weather.example/intro.vxml">
     Weather
  </choice>
  <choice next="http://www.news.example/news.vxml">
     News
  </choice>
  <noinput>Please say one of <enumerate/></noinput>
</menu>
</vxml>
```

Computer:	Say one of: Sports; Weather; News.
Human:	Astrology
Computer:	I did not understand what you said. *(a platform-specific default message.)*
Computer:	Say one of: Sports; Weather; News.
Human:	Sports
Computer:	*(proceeds to http://www.sports.example/start.vxml)*

SRGS (BNF, XML)

https://www.w3.org/TR/speech-grammar/

```
#ABNF 1.0 ISO-8859-1;

// Default grammar language is US English
language en-US;

// Single language attachment to tokens
// Note that "fr-CA" (Canadian French) is applied to only
//  the word "oui" because of precedence rules
$yes = yes | oui!fr-CA;

// Single language attachment to an expansion
$people1 = (Michel Tremblay | André Roy)!fr-CA;

// Handling language-specific pronunciations of the same word
// A capable speech recognizer will listen for Mexican Spanish and
//   US English pronunciations.
$people2 = Jose!en-US | Jose!es-MX;

/**
 * Multi-lingual input possible
 * @example may I speak to André Roy
 * @example may I speak to Jose
 */
public $request = may I speak to ($people1 | $people2);
```