Pushing the frontiers of speech processing — What does it take to tackle new languages and domains?

Florian Metze, Samuel Thomas,
Bhuvana Ramabhadran, and Brian Kingsbury

Carnegie Mellon University and IBM

For the most recent version of this presentation

https://github.com/bedk/IS2016-Tutorial/blob/master/main.pdf

Florian Metze

- Associate Research Professor at Carnegie Mellon University (LTI/SCS)
- End-to-end Speech Recognition
- Articulatory Features for Speech Recognition
- Multi-media Analysis
- `fmetze@cs.cmu.edu`

Samuel Thomas

- Researcher in the IBM Watson Group
- Automatic Speech Recognition
- Feature Engineering and Acoustic Modeling
- sthomas@us.ibm.com

Bhuvana Ramabhadran

- Research Manager in the IBM Watson Group
- Automatic Speech Recognition
- Deep Learning for Acoustic and Language Modeling
- Keyword Search
- bhuvana@us.ibm.com

Brian Kingsbury

- Research Scientist in the IBM Watson Group
- Deep Learning
- Automatic Speech Recognition
- Keyword Search
- bedk@us.ibm.com

Outline (3 hours)

1. Introduction
2. Tackling a New Language
3. Tackling a New Domain
4. Research Topics, Challenges, and New Ideas
5. End-to-End Systems
6. Virtual Machines and Tools
7. Conclusions

Introduction

Introduction

Outline

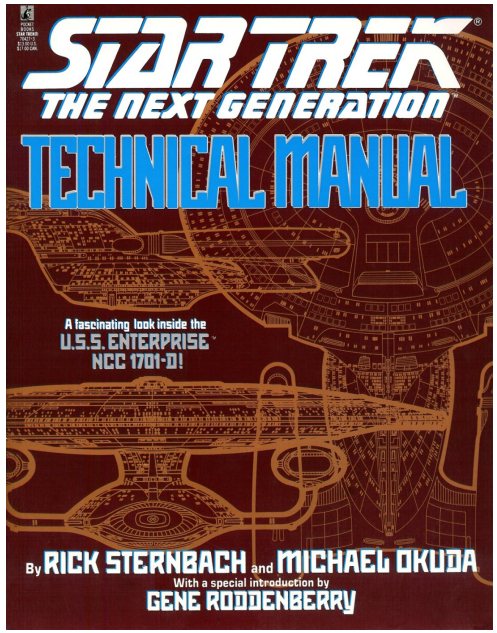
1. The shift in speech based user interfaces
2. Building applications on the information rich speech signal
3. The Speech Recognition Case Study
   - From rule-based systems to data-driven system
   - Impact of speech recognition technologies across languages and domains
   - What is under the hood for speech recognition technologies?
   - Building various ASR module and the impact of data
4. Building ASR Systems in New Languages
   - Building from ASR systems from scratch
   - Is there room for sharing data from other languages?
5. Building ASR Systems in New Domain
   - Adaptation of an existing ASR system

Speech based user interfaces

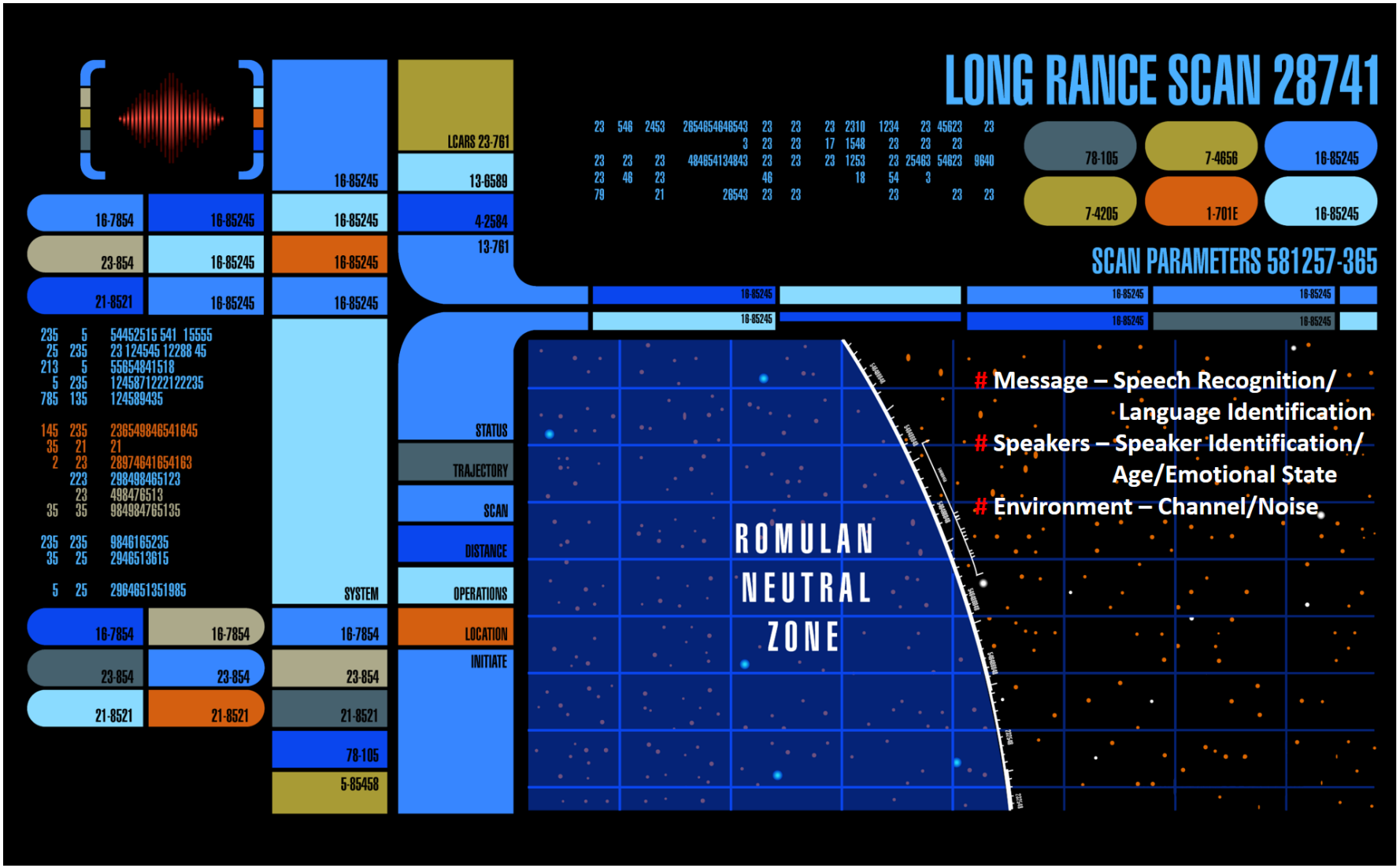
Star Trek's Universal Translator [Wikipedia] - . . . used in the late 22nd century on Earth for the instant translation of well-known Earth languages. Gradually, with the removal of language barriers, Earth's disparate cultures came to terms of universal peace. Translations of previously unknown languages, such as those of aliens, required more difficulties to be overcome.

Speech based user interfaces

Star Trek's The Next Generation Technical Manual [Wikipedia] - . . . the Universal Translator is an "extremely sophisticated computer program" which functions by analyzing the patterns of an unknown foreign language, starting from a speech sample of two or more speakers in conversation. The more extensive the conversational sample, the more accurate and reliable is the "translation matrix," enabling instantaneous conversion of verbal utterances or written text between the alien language and Federation Standard.



Building applications on the speech signal



The march to becoming ubiquitous?

- 1877 Thomas Edison's phonograph
- 1936 First electronic speech synthesizer - the Voder
- 1962 IBM's Shoebox - understands up to 16 spoken
- 1976 DARPA program - CMU's 1K word recognizer - Harpy
- 1980 HMMs - IBM's 20K word recognizer - Tangora
- 1990 Dragon Dictate, consumer based speech recognition
- 1993 CMU's Sphinx-II LVCSR system
- 1996 IBM's MedSpeak commercial LVCSR system
- 2007 Microsoft's Windows Vista with speech recognition
- 2008 Google's Voice Search for iPhone
- 2011 Apple's Siri
- 2014 Microsoft's Cortana/Amazon's Echo

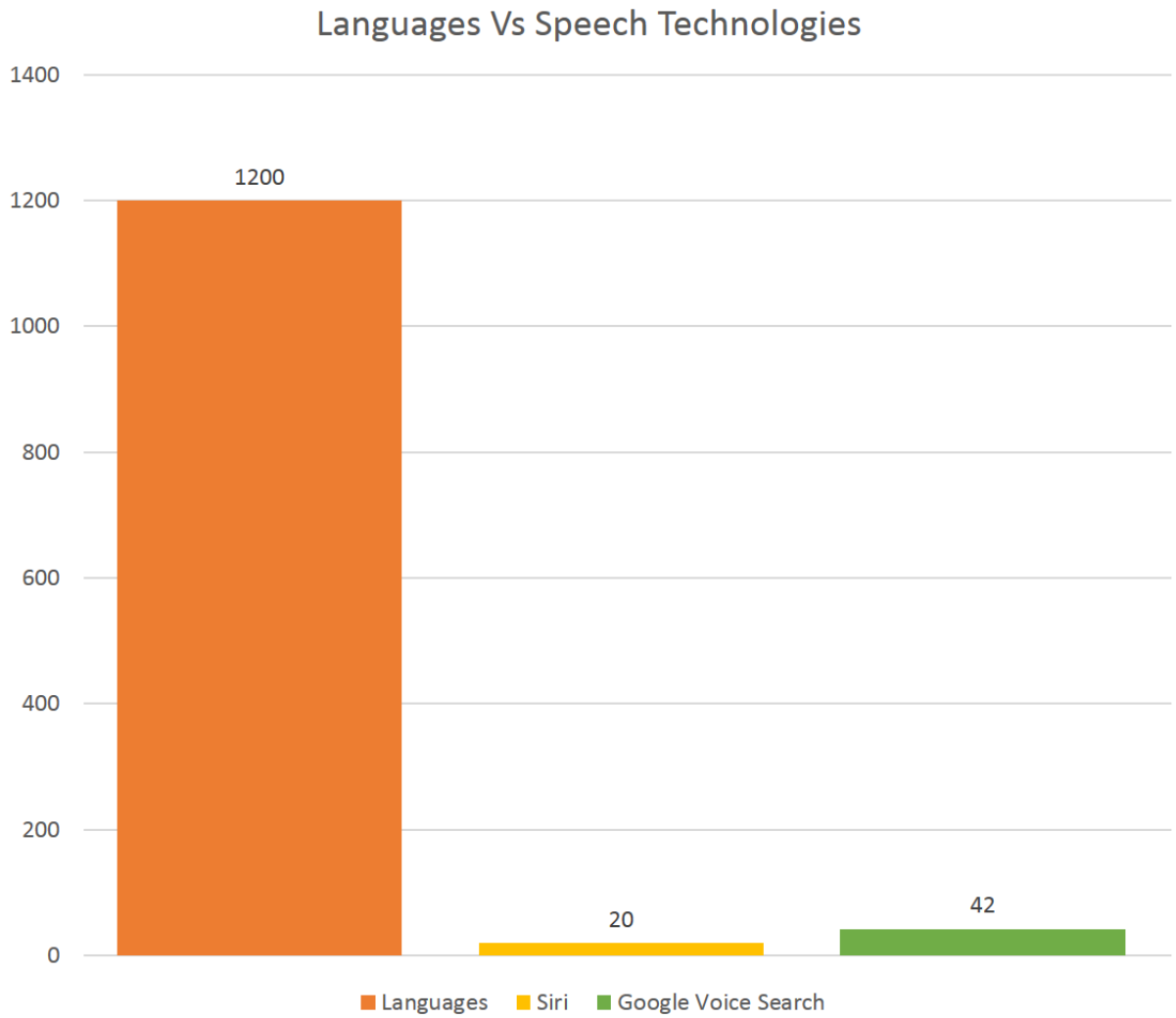
In how many different ways can we speak?

Distribution of world languages by number of first-language speakers

Population range	Living languages			Number of speakers		
	Count	*Percent*	*Cumulative*	*Total*	*Percent*	*Cumulative*
100,000,000 to 999,999,999	8	0.1	0.1%	2,614,031,200	40.17718	40.17718%
10,000,000 to 99,999,999	84	1.2	1.3%	2,608,411,777	40.09081	80.26798%
1,000,000 to 9,999,999	306	4.3	5.6%	917,412,798	14.10047	94.36845%
100,000 to 999,999	944	13.3	18.9%	297,092,808	4.56626	98.93471%
10,000 to 99,999	1,808	25.5	44.4%	61,213,595	0.94084	99.87555%
1,000 to 9,999	1,979	27.9	72.3%	7,614,348	0.11703	99.99258%
100 to 999	1,070	15.1	87.3%	469,345	0.00721	99.99980%
10 to 99	337	4.7	92.1%	12,753	0.00020	99.99999%
1 to 9	132	1.9	94.0%	536	0.00001	100.00000%
0	220	3.1	97.1%	0	0.00000	100.00000%
Unknown	209	2.9	100.0%			
Totals	7,097	100.0		6,506,259,160	100.00000	

Do speech technologies stand tall?

Less than 4% of the language space has been covered!



For a given language, New Speech Applications = New Domains?

1. Applications
   - 1.1 Mobile phone applications - voice search
   - 1.2 IVR applications/Call centers
   - 1.3 Dictation - Medical Transcription
   - 1.4 Surveillance - Military/Law enforcement
   - 1.5 In-car/at-home appliances
   - 1.6 Education/Accessibility
   - 1.7 Transcription - Court reporting
   - 1.8 Robotics/Gaming
2. Noise distortions to speech input of existing applications
   - 2.1 Additive Noise
   - 2.2 Convolutive Noise
3. Every new speech recognition deployment!

Dealing with New Languages and Domains

New Languages

1. Every language is unique although they might share many constructs with other languages
2. Separate resources specific to the language for building ASR systems are required, possibility to share resources across languages
3. Building a new ASR system from scratch

New Domains

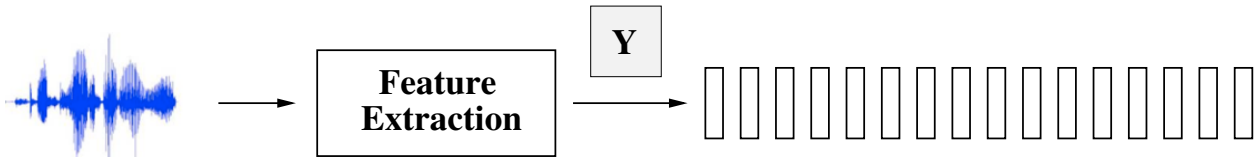
1. Domains might have specific constructs but usually involves tailoring existing resources of a language
2. Resources from the same languages can be shared
3. Usually an ASR adaptation problem rather than building from scratch

What's under the hood for ASR?

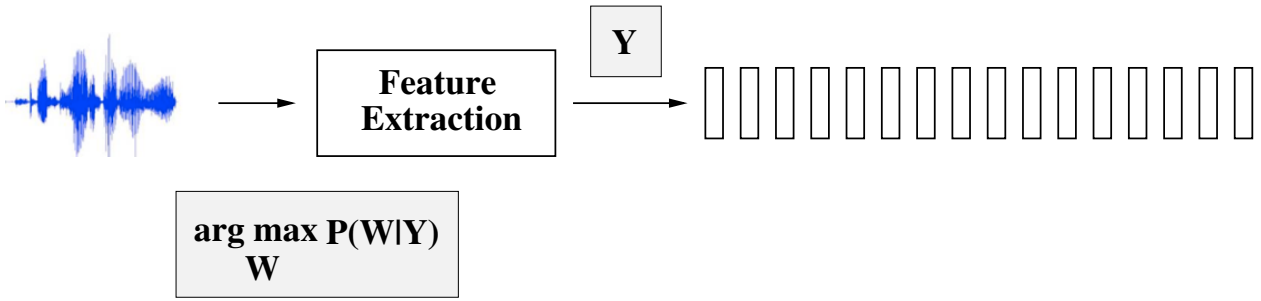
1. Automatic speech recognition is the process of transcribing speech into text
2. Speech recognition systems solve this task in a probabilistic setting using five key components: a feature extraction module, an acoustic model, a pronunciation dictionary, a language model and a search module.

What's under the hood for ASR?

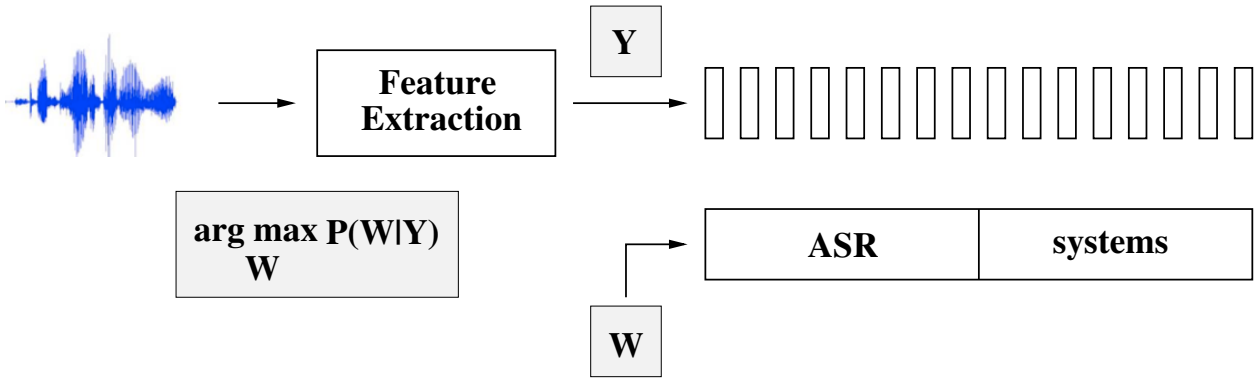
What's under the hood for ASR?



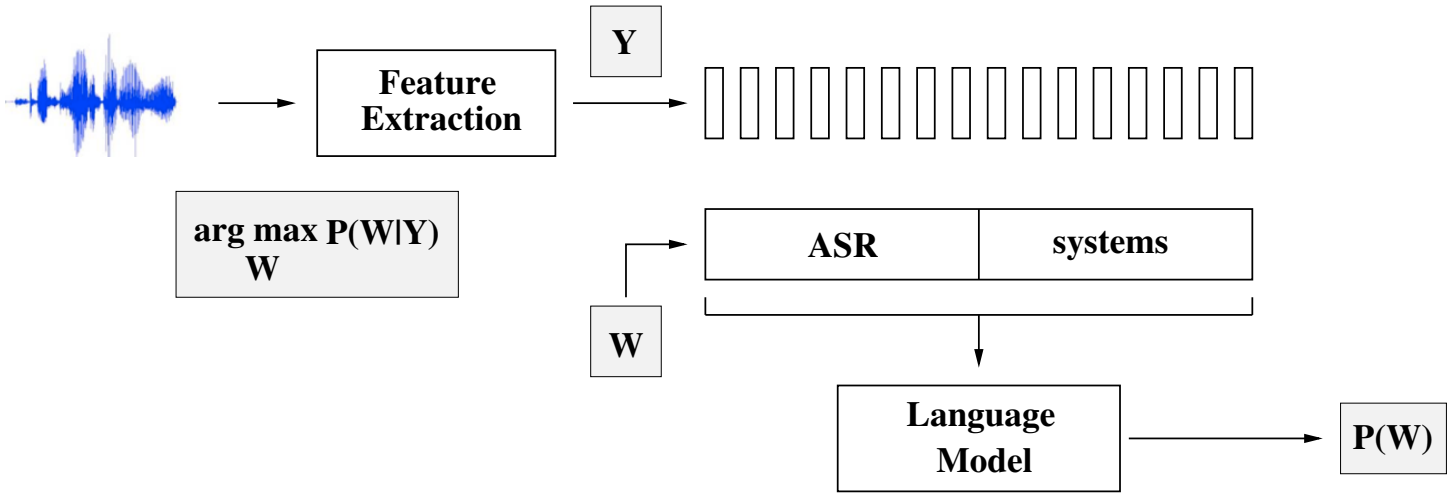
What's under the hood for ASR?



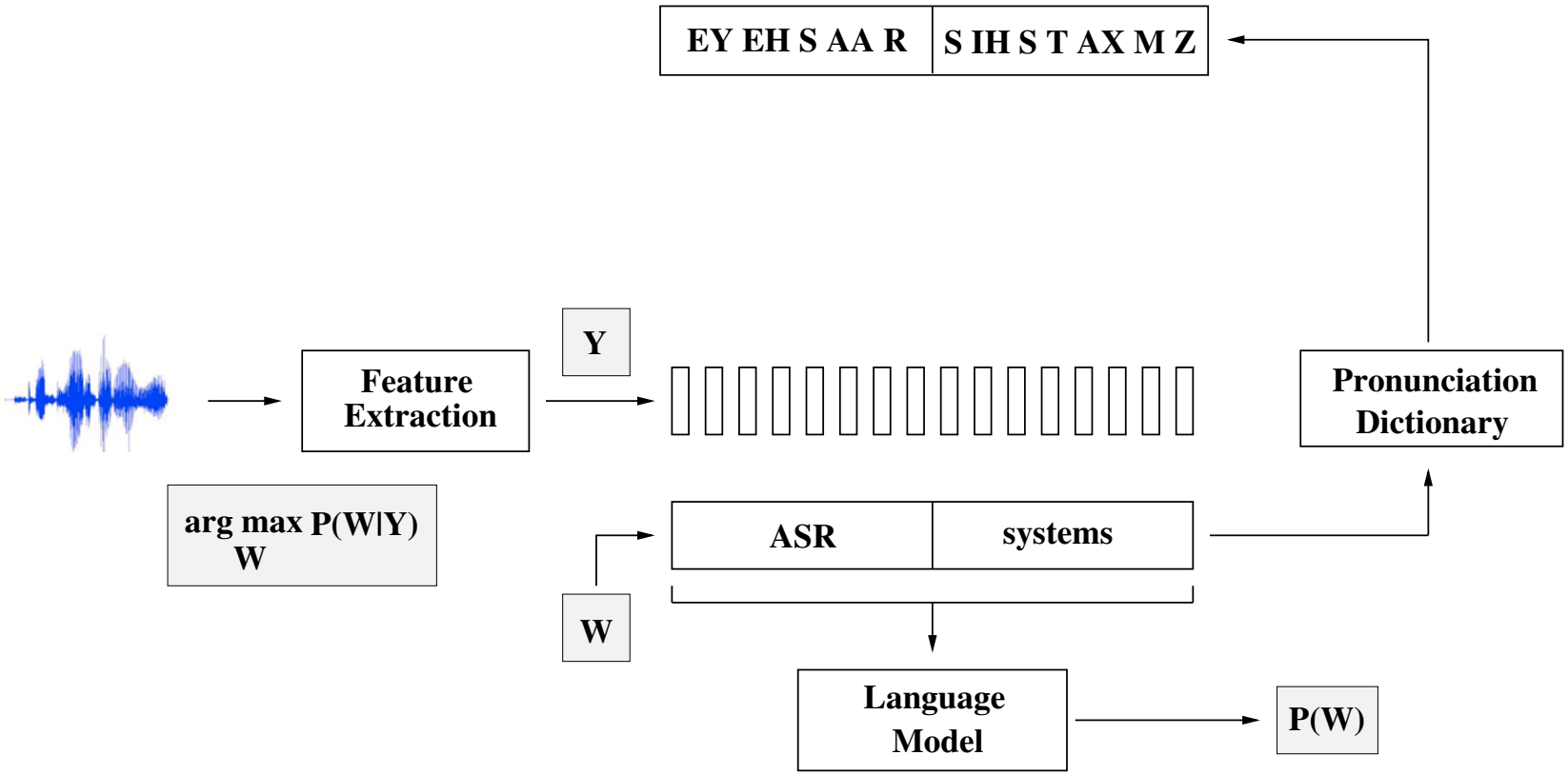
What's under the hood for ASR?



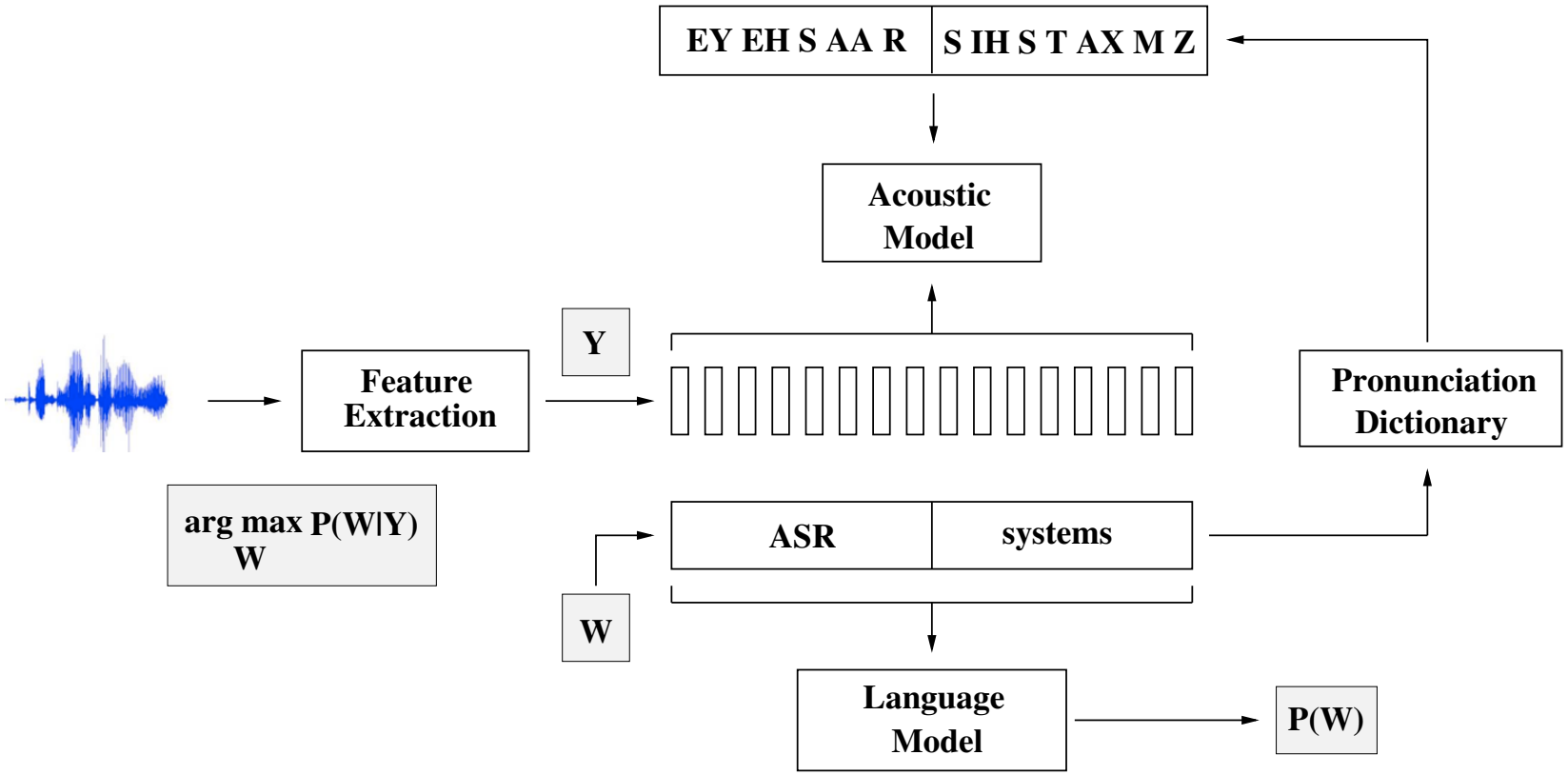
What's under the hood for ASR?



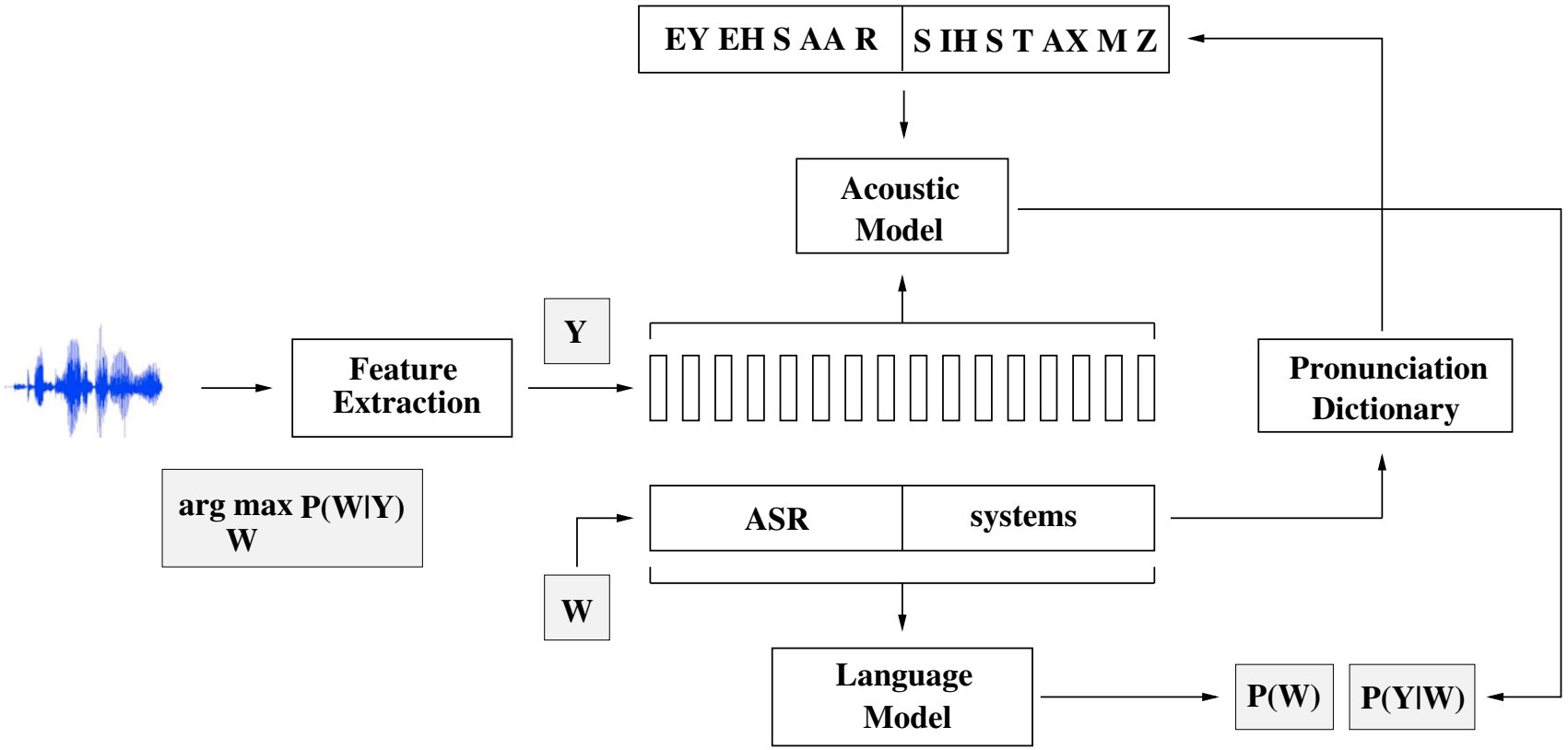
What's under the hood for ASR?



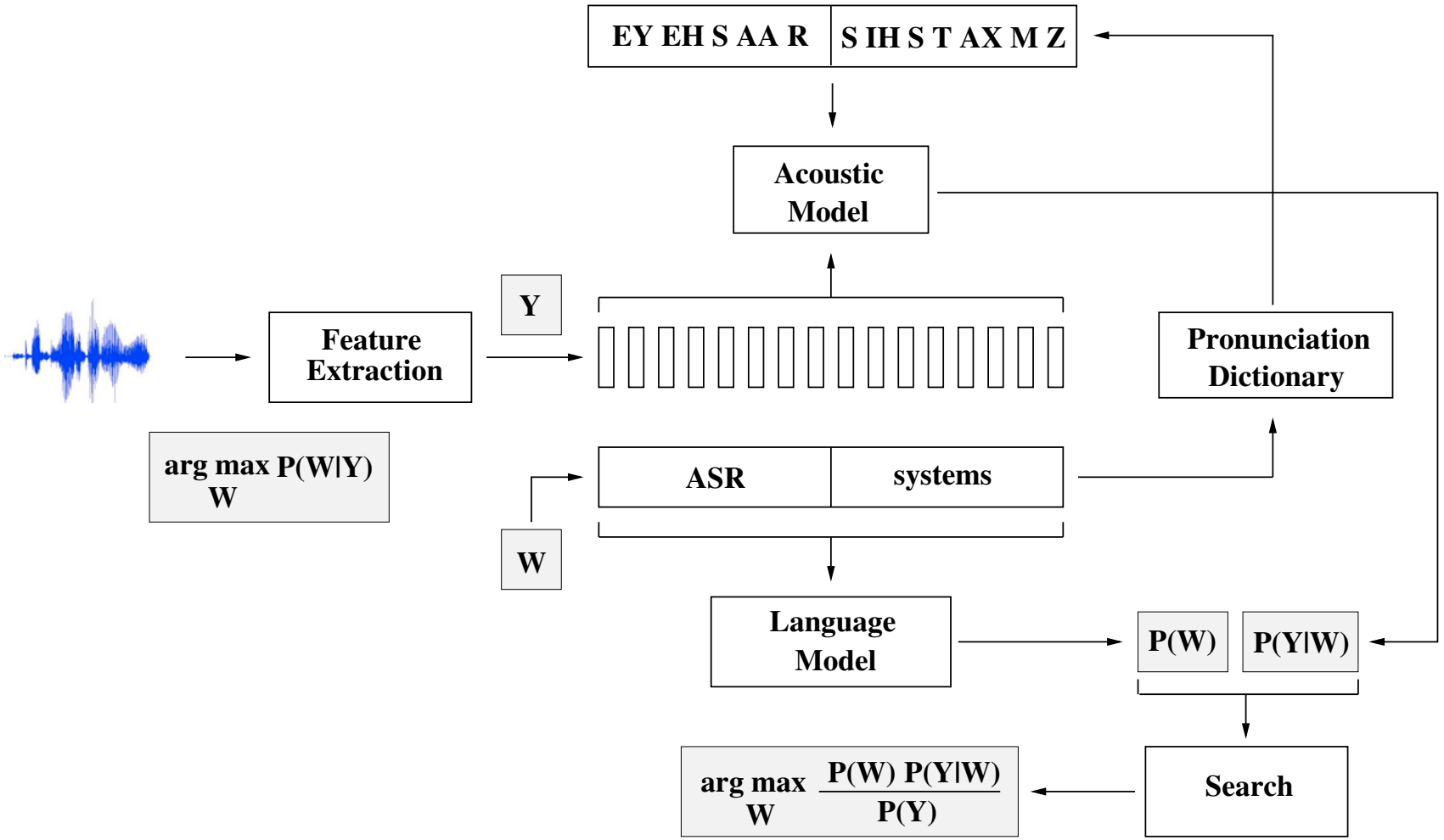
What's under the hood for ASR?



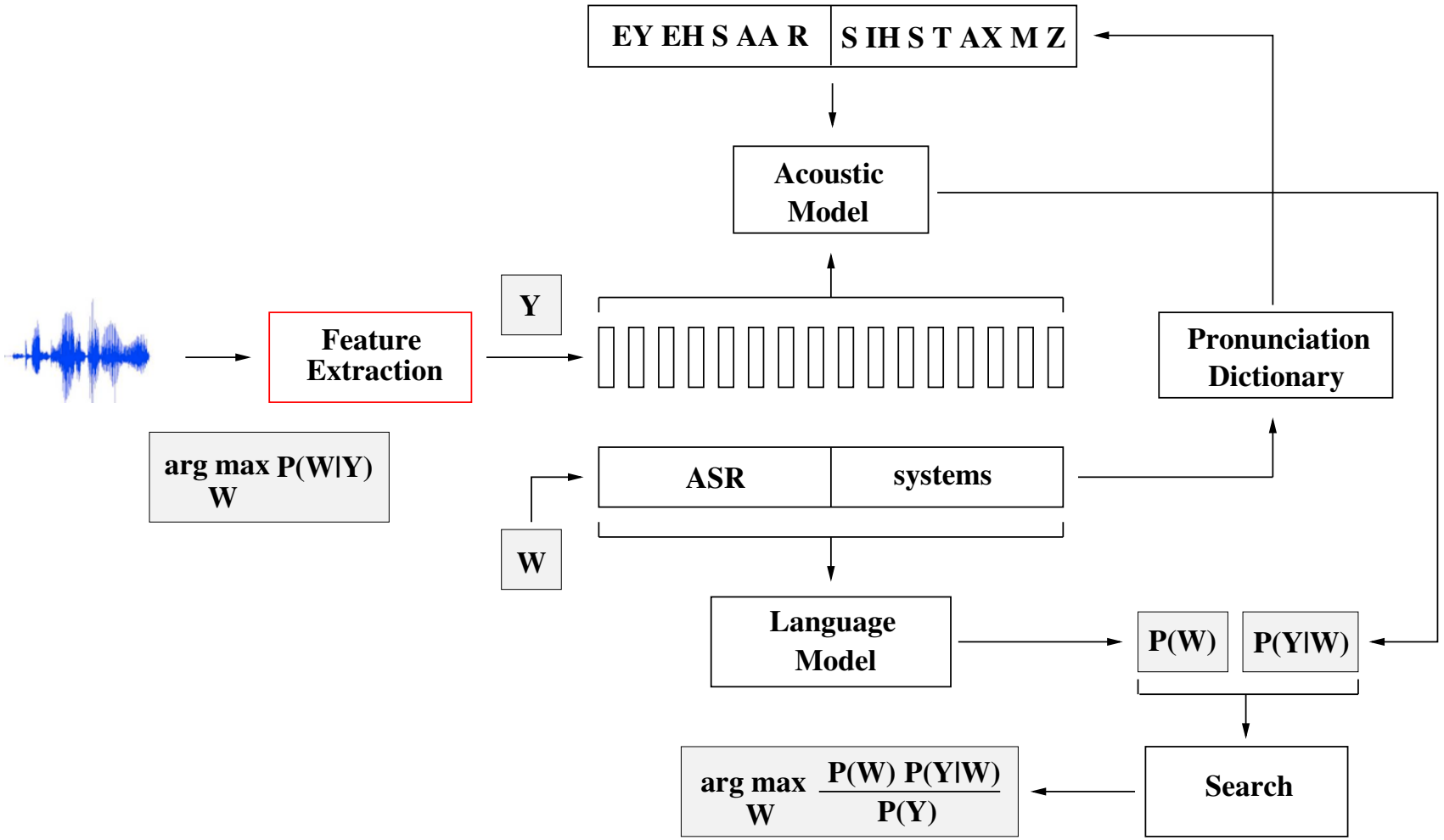
What's under the hood for ASR?



What's under the hood for ASR?



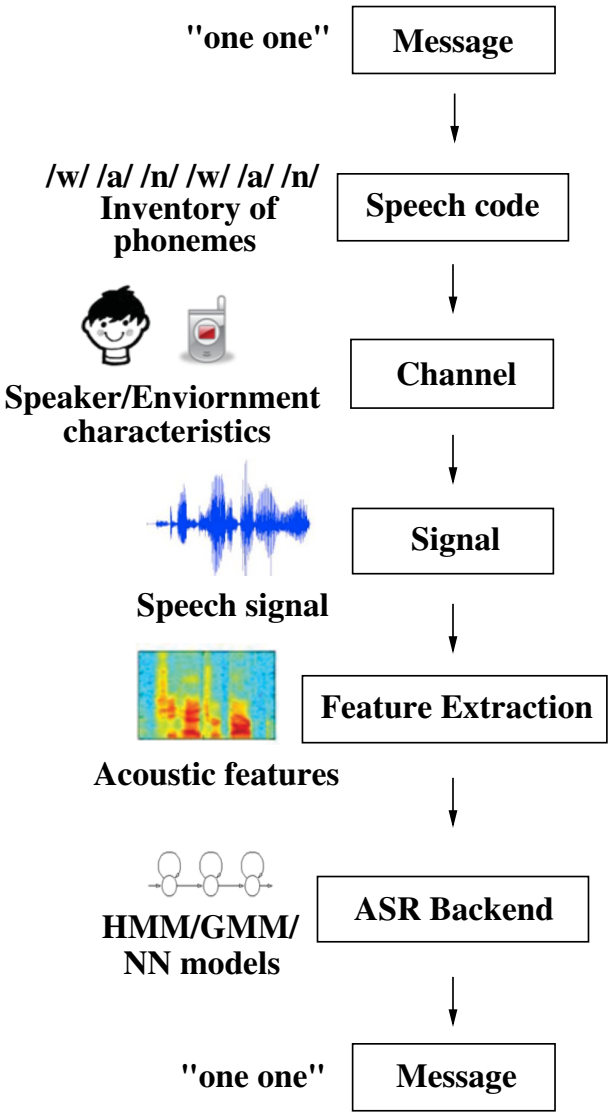
What's under the hood for ASR?



Feature Extraction - Introduction

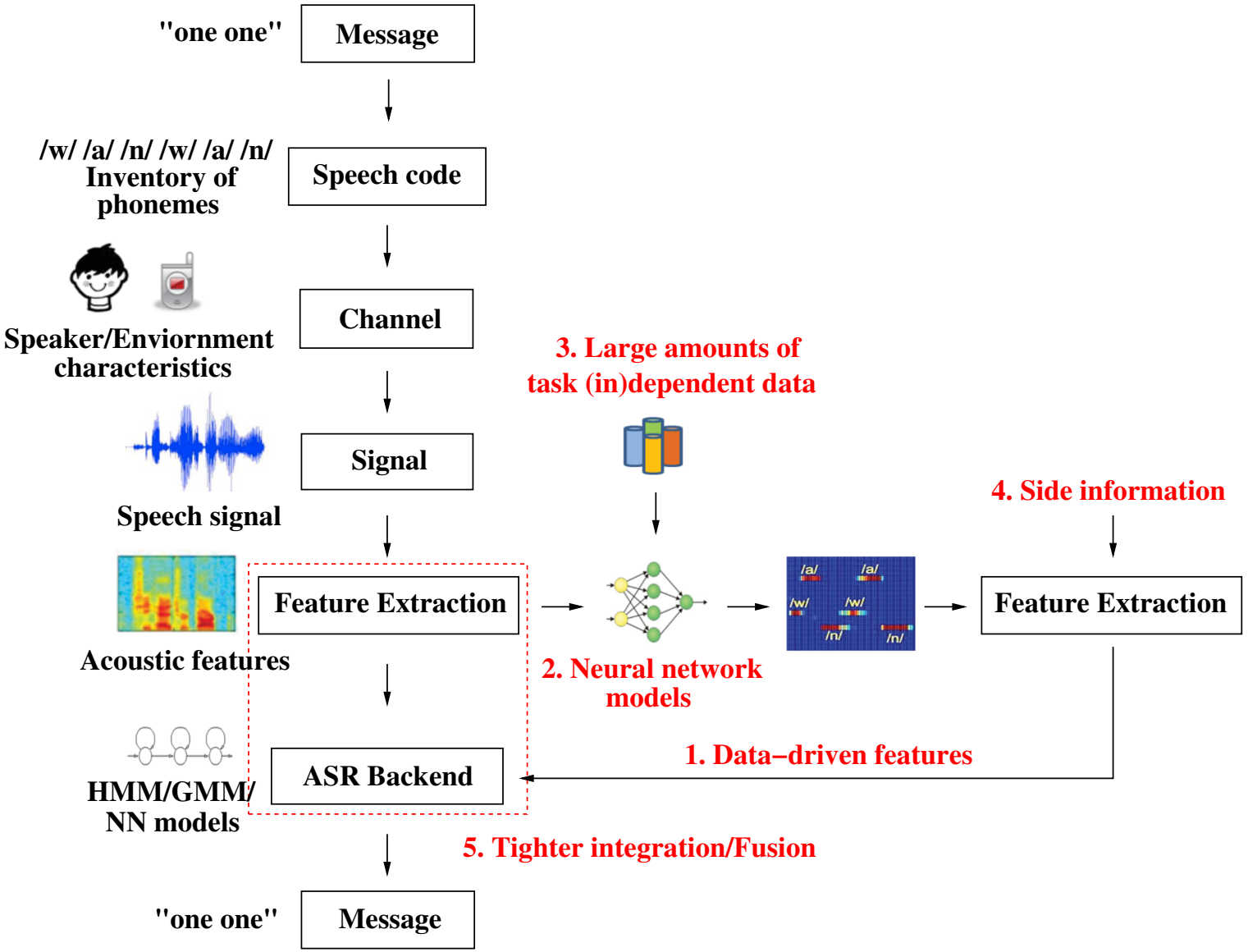
1. Why not use the speech signal directly for pattern recognition?
   - 1.1 Speech is a time-varying non-stationary signal
   - 1.2 Information may lie in a small portion of signal
   - 1.3 May contain irrelevant information for the application
   - 1.4 Presence of noise and other distortions
   - 1.5 Size and dimensionality of the data
2. Need to transform the signal into lower dimensional descriptors called features.

Feature Extraction - "Past" Techniques

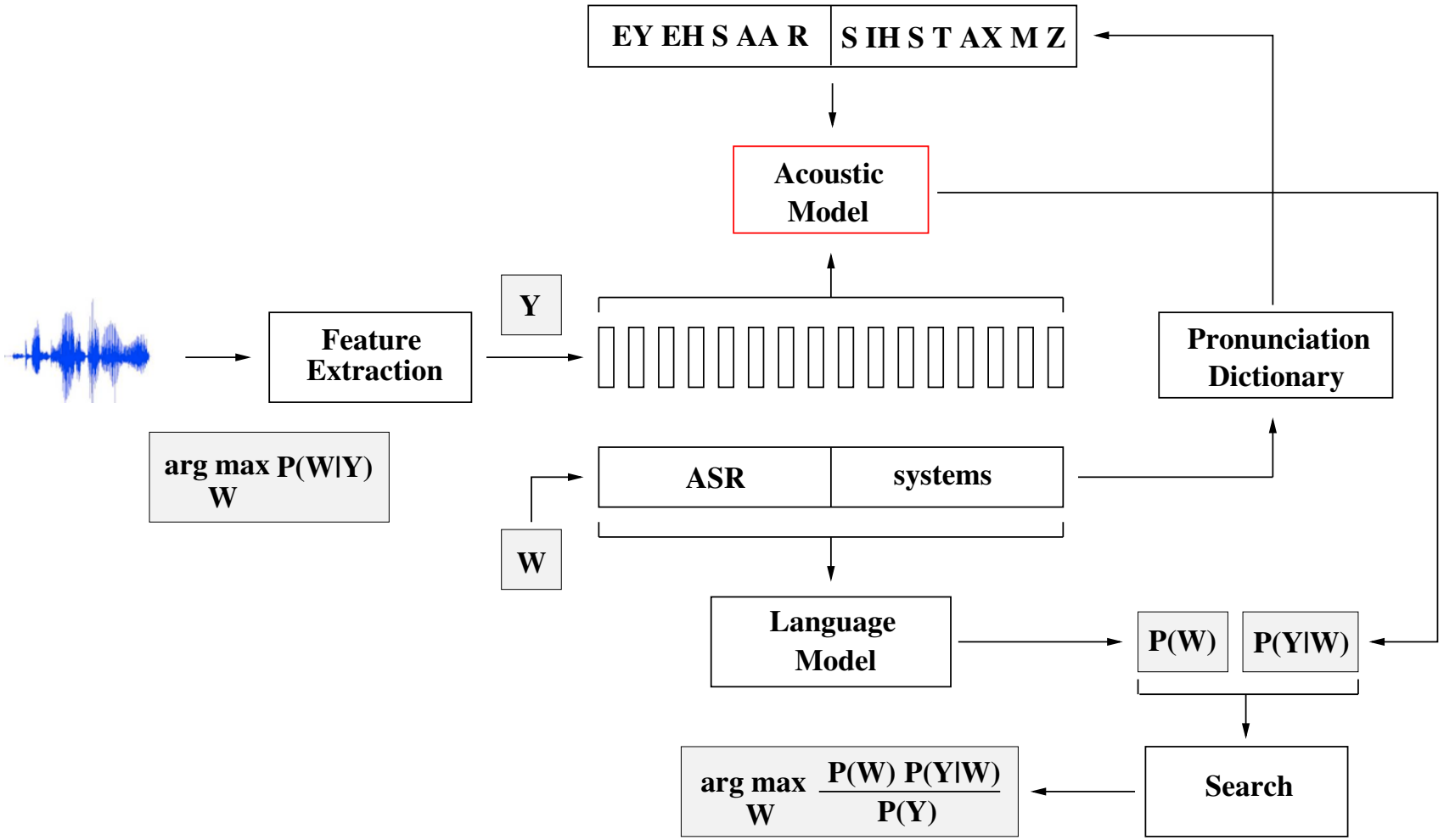


1. Speech Coding Inspired Features - Mel Spectrogram and Linear Prediction
2. Speech Synthesis Inspired Features - Pitch and Prosody
3. Long Contextual Features - Delta Processing, RASTA Filtering and Modulation Features
4. Normalization Techniques - Cepstral Mean Normalization and Spectral Subtraction

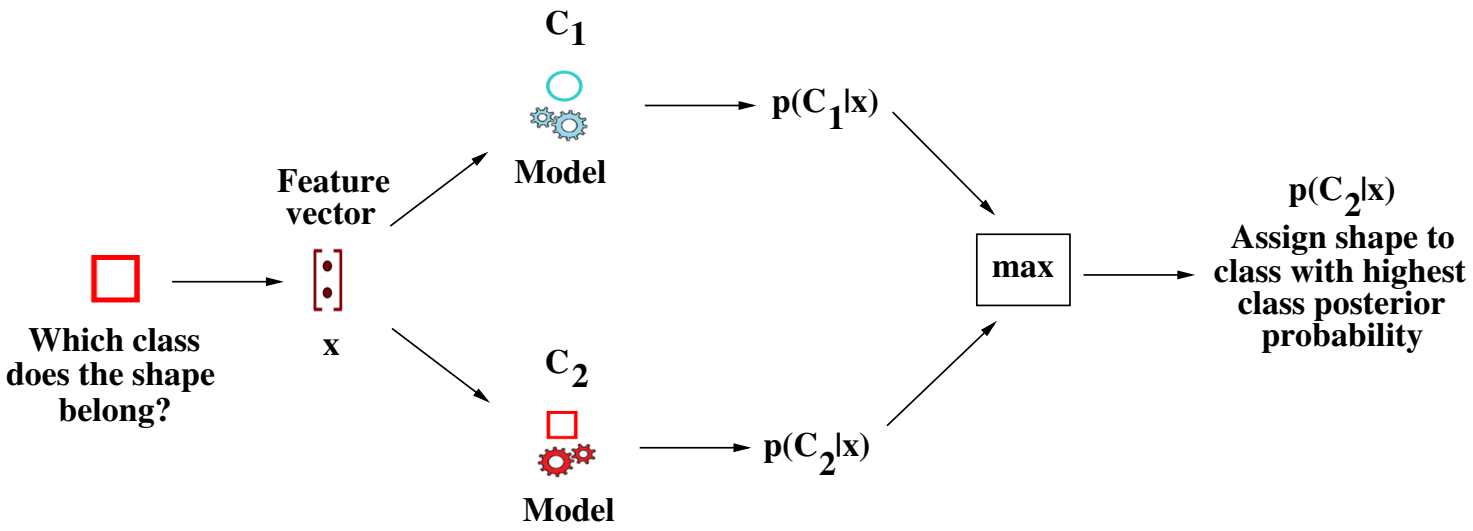
Feature Extraction - Current Techniques



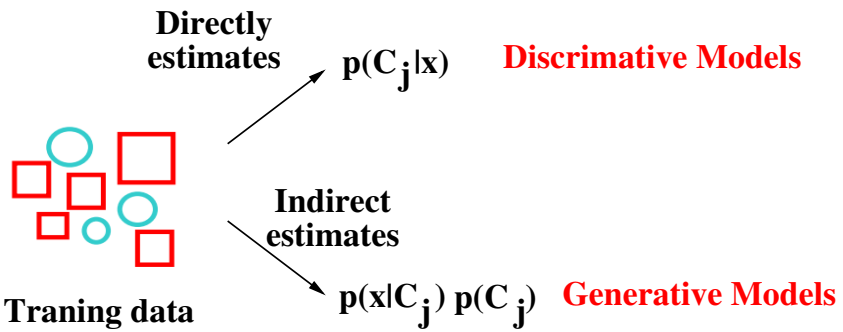
What's under the hood for ASR?



Acoustic Modeling - Introduction



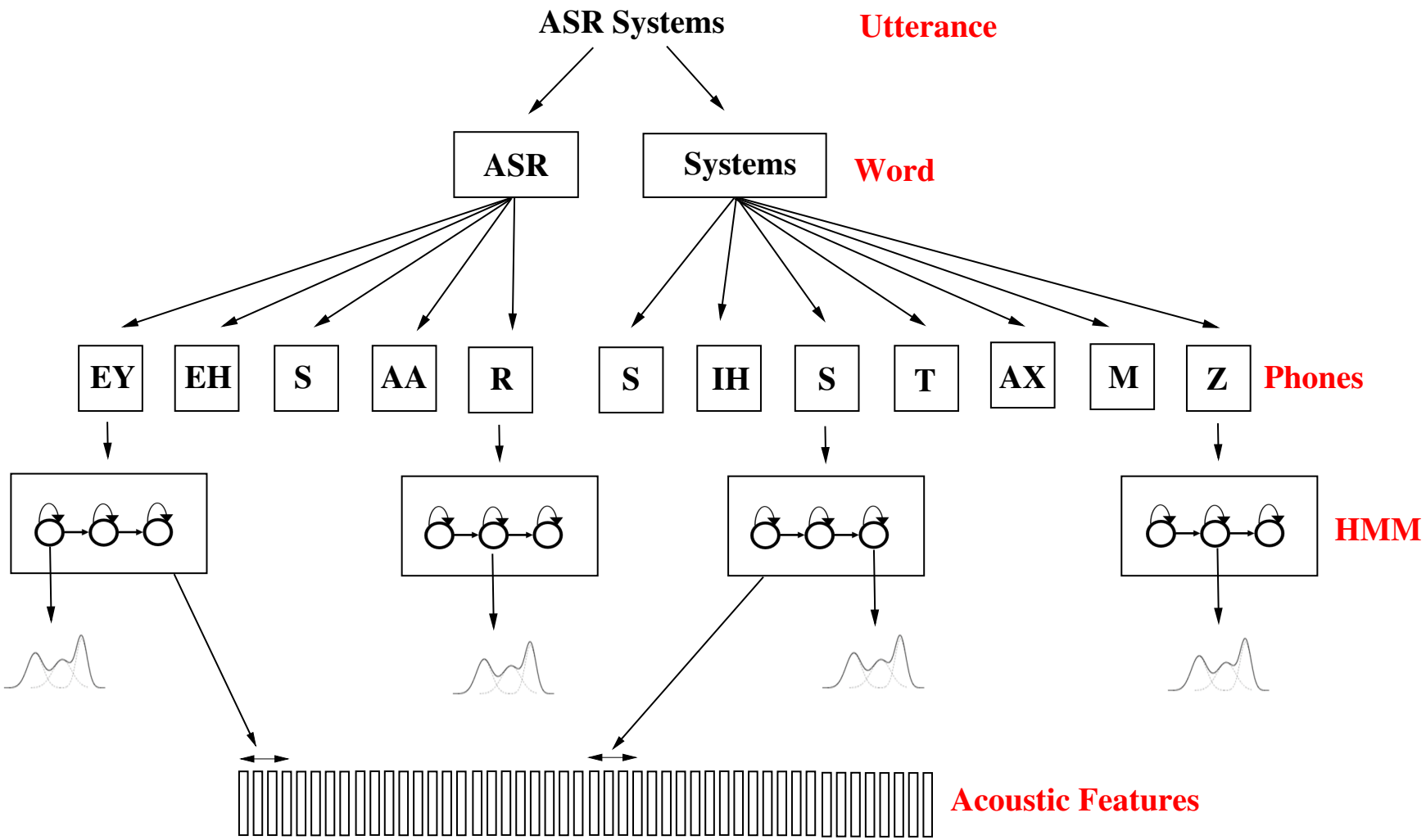
$$p(C_j|x) = \frac{p(x|C_j)\,p(C_j)}{p(x)}$$



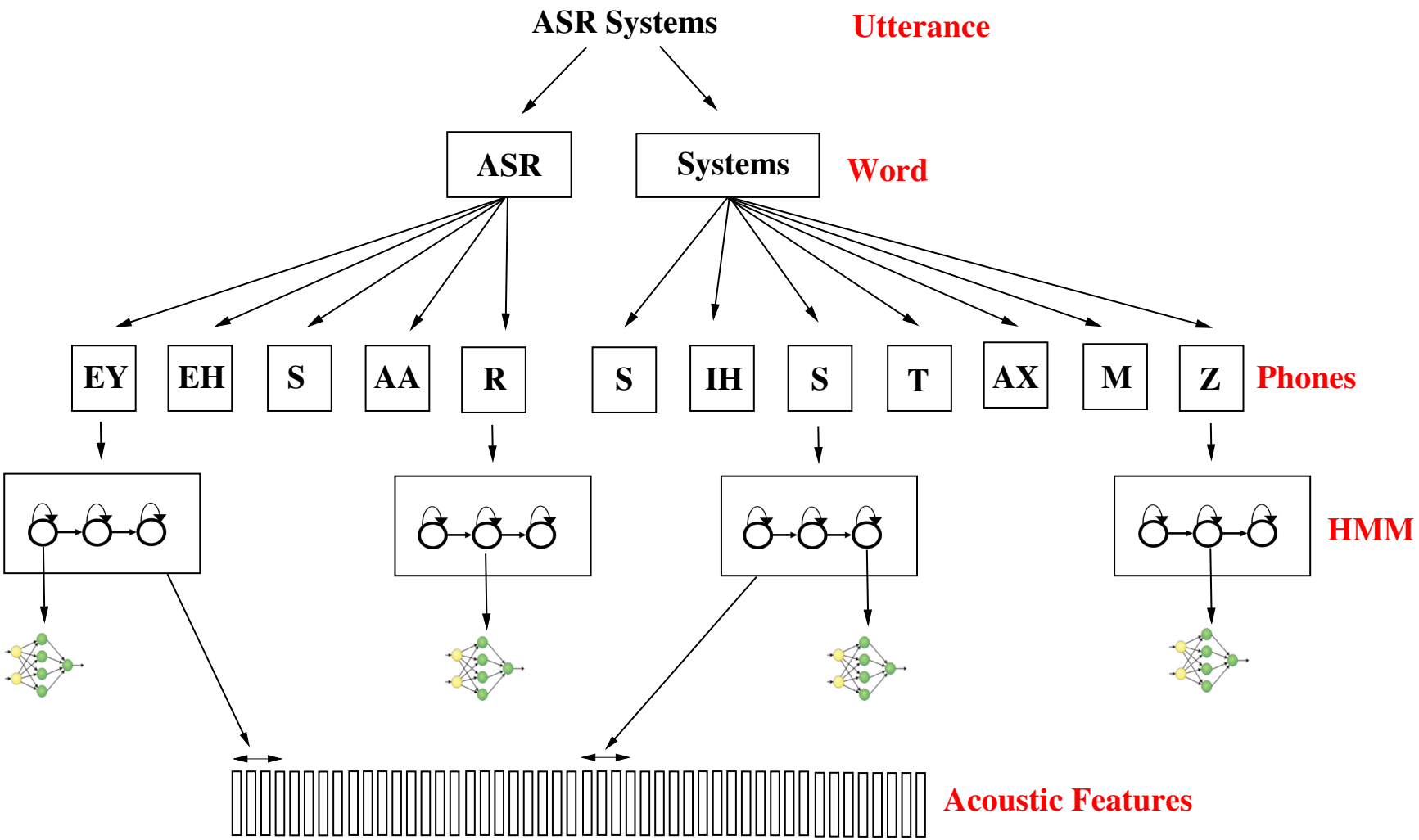
Acoustic Modeling - "Past" Techniques

1. The goal of the AM is to provide a method of calculating the likelihood of any feature vector sequence Y given a word w
2. The basic sequence model -
   - 2.1 To estimate this likelihood word sequences are decomposed into basic sound units called phones
   - 2.2 Each individual phone is represented using a Hidden Markov Model (HMM) - a state machine with a number of states connected with arcs in a left-right topology.
   - 2.3 Each state models the distribution of acoustic vectors with Gaussian mixture models.
   - 2.4 HMM algorithms: Likelihood computation (forward algorithm), most probable state sequence (Viterbi algorithm), estimating parameters (EM algorithm)
3. Improvements to the basic model -
   - 3.1 Modeling - Context-dependent models, Tied mixture models, state clustering, phonetic decision trees
   - 3.2 Training criteria - MLE, MMIE, MPE for training HMM-GMM based systems

Acoustic Modeling - "Past" Techniques



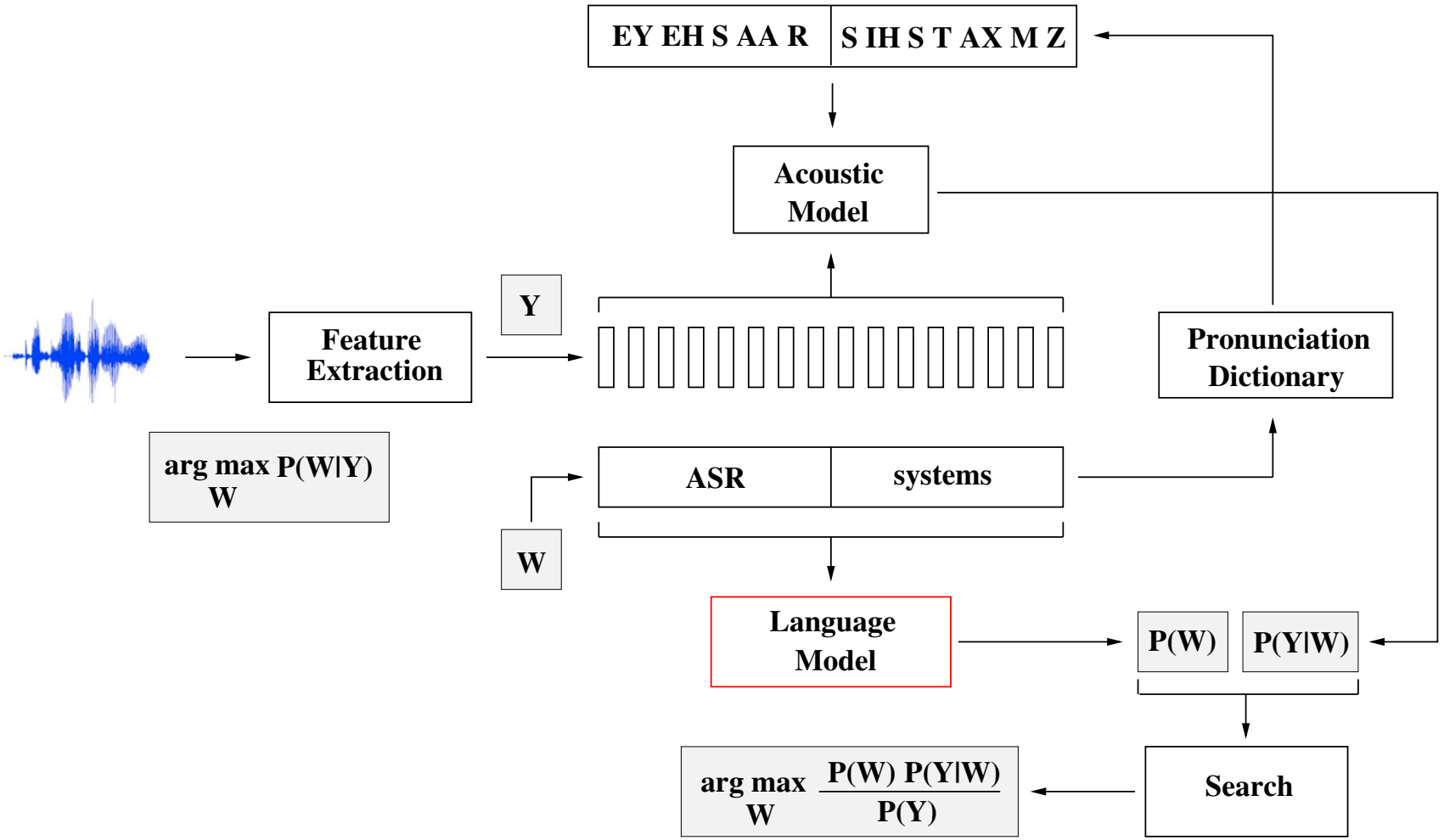
Acoustic Modeling - Current Techniques



Acoustic Modeling - Current Techniques

1. The basic model -
   - 1.1 Train neural networks to estimate posterior probabilities of speech classes
   - 1.2 Instead of GMM based likelihood estimates use scaled likelihoods based off neural network posterior probabilities
   - 1.3 Trained to discriminate between output classes with a cross-entropy based training criteria.
2. Improvements to the basic model -
   - 2.1 Deeper networks, output states corresponding to context dependent states,
   - 2.2 Various configurations - CNNs, RNNs, LSTMs
   - 2.3 Modeling using larger context and correlated features
   - 2.4 Tighter integration of feature extraction and acoustic modeling

What's under the hood for ASR?



Language Modeling - Introduction

1. Estimate the prior probability of a word sequence $P(W)$
2. Provides the ASR system with information about contextual relationships at a much higher word level
3. Helps constrain the search space by helping disambiguate between word sequences that sound acoustically similar - "recognize speech" vs. "wreck a nice beach"

Language Modeling - "Past" Techniques

1. The basic model -
   - 1.1 Simple n-grams based language models
   - 1.2 Probability of a word depends on the word and preceding n-1 words
   - 1.3 For a word sequence $W = w_1, w_2, \ldots, w_N$,

$$P(W) = P(w_1)P(w_2|w_1)P(w_3|w_1, w_2)\ldots P(w_N|w_1, w_2, \ldots w_{N-1})$$

   - 1.4 A bigram approximation limits the context to one word

$$P(W) \simeq P(w_1)P(w_2|w_1)P(w_3|w_2)\ldots P(w_N|w_{N-1})$$

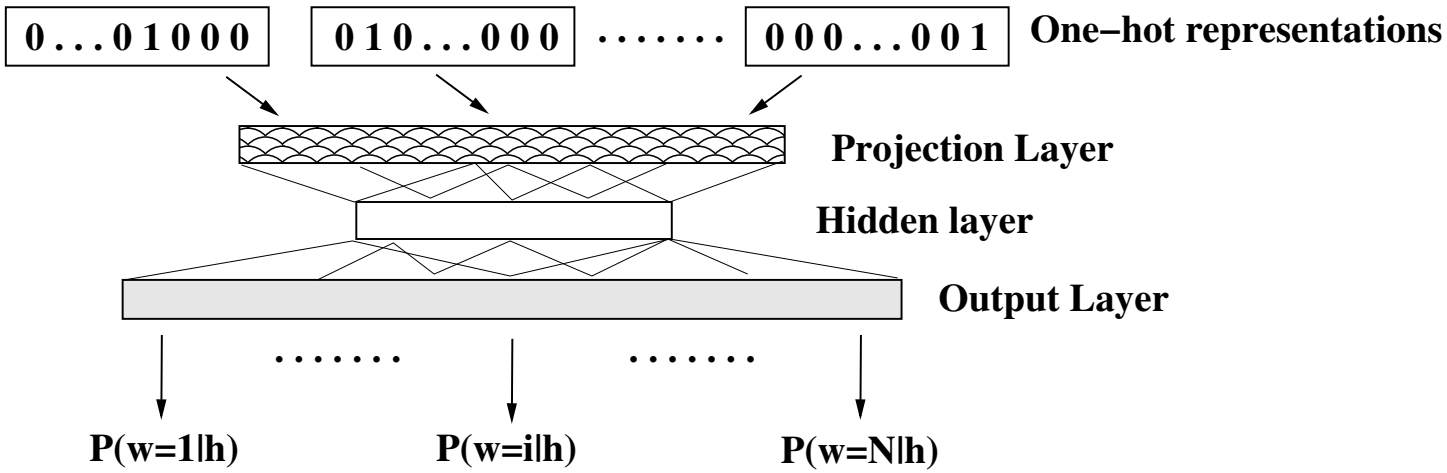
   - 1.5 Maximum likelihood estimates for conditional probabilities using counts

$$P(w_i|w_j) = \frac{c(w_j, w_i)}{c(w_j)}$$

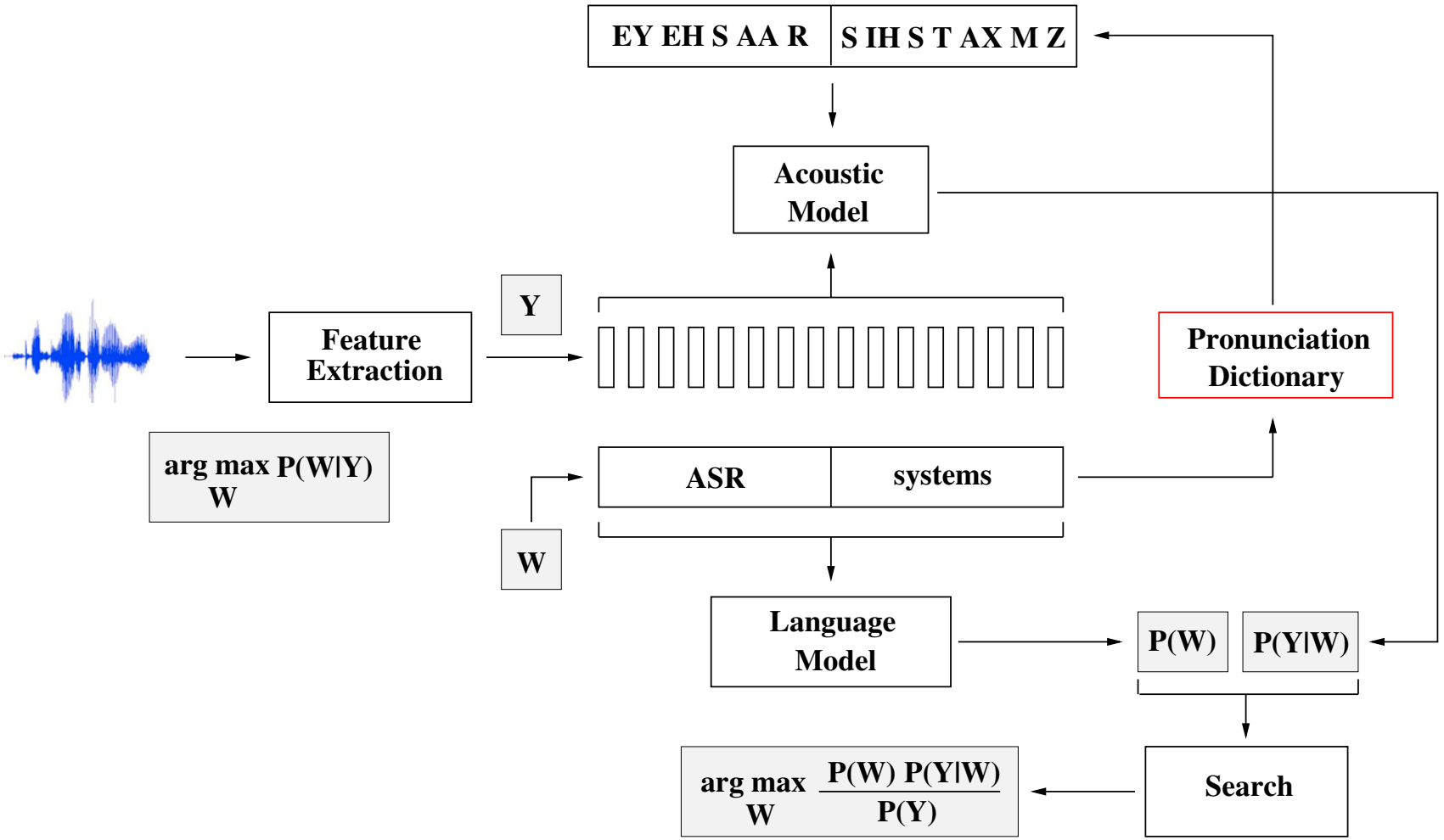
   - 1.6 Unseen ngrams accounted for using smoothing/discounting/backoff
2. Improvements to the basic model -
   - 2.1 Class-based n-grams, maximum entropy language models

Language Modeling - Current Techniques

Neural network based language models



What's under the hood for ASR?



Pronunciation Lexicon

1. Provides pronunciations of words linking the AM and LM
2. Typically words are represented phonetically

   vases(01) V EY Z IH Z

3. Pronunciation variants also need to be added

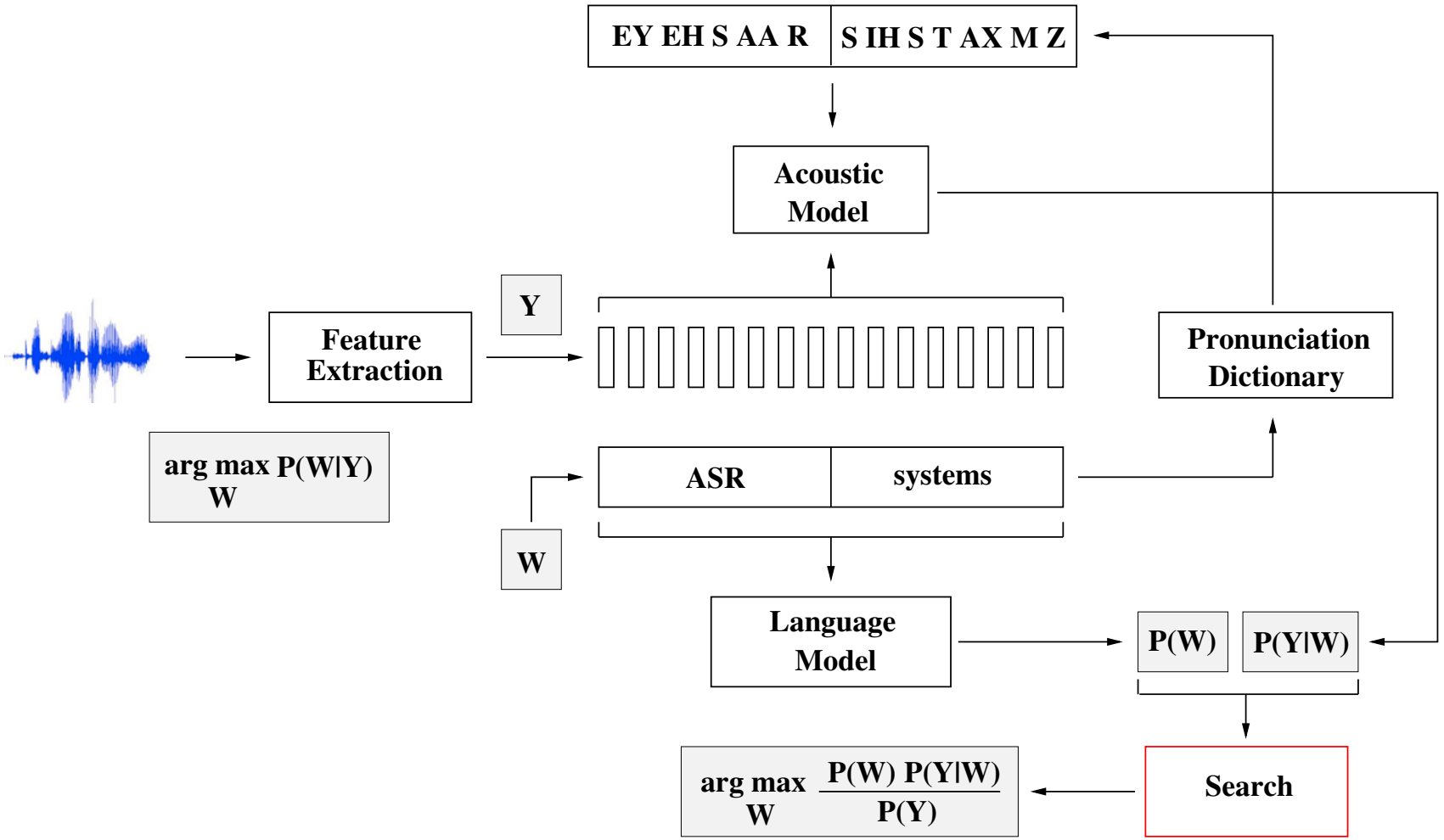
   vases(01) V EY Z IH Z
   vases(02) V AA Z IH Z
   vases(03) V EY S IH Z

4. Expensive since pronunciations are created by human experts

Pronunciation Lexicon

1. The basic model -
   - 1.1 Dictionary created by human experts
   - 1.2 Grapheme-to-phoneme models - learn pronunciations of words based on initial dictionary
2. Morphologically rich languages - modeling at morph level
   - 2.1 Compounding (eg. German)
   - 2.2 Highly inflected languages (eg. Slavic languages)
   - 2.3 Inflection and compounding (eg. Finnish)
3. Graphemic dictionaries - modeling at the character level - bypass the need for pronunciations

What's under the hood for ASR?



The Decoder

1. HMM-based speech recognition - Viterbi decoding - efficient dynamic programming based search for optimal exhaustive search
2. Search for large vocabulary task is complex -
   - 2.1 Cross-word triphones and word boundaries
   - 2.2 Huge search spaces with long-span language models
3. Improved techniques for search -
   - 3.1 Beam pruning
   - 3.2 Multipass search
   - 3.3 Dynamic networks
   - 3.4 Weighted finite state transducers

New Languages and Domains - More specifics!

New Languages

1. Every language is unique although they might share many constructs with other languages
2. Separate resources specific to the language for building ASR systems are required
   - 2.1 Feature extraction
   - 2.2 Acoustic models
   - 2.3 Language models
   - 2.4 Pronunciation lexicons
3. Building a new ASR system from scratch

New Domains

1. Domains might have specific constructs but usually involves tailoring existing resources of a language
2. Resources from the same languages can be shared
   - 2.1 Feature extraction
   - 2.2 Acoustic models
   - 2.3 Language models
   - 2.4 Pronunciation lexicons
3. Usually an ASR adaptation problem rather than building from scratch

Tackling New Languages

Tackling New Languages

Outline

1. IARPA Babel
2. Audio Keyword Search
3. What Language Characteristics Matter?
4. A Recipe for a New Language

The IARPA Babel Program

"...to rapidly develop speech recognition capability for keyword search in a previously unstudied language, working with speech recorded in a variety of conditions with limited amounts of transcription."

Rapid Development

Time allowed for surprise language model building

Period	Time
1	4 weeks
2	3 weeks
3	2 weeks
4	1 week

The IARPA Babel Program

"...to rapidly develop speech recognition capability for keyword search in a previously unstudied language, working with speech recorded in a variety of conditions with limited amounts of transcription."

Babel Languages

Period 1	Period 2	Period 3	Period 4
Cantonese	Assamese	Kurmanji Kurdish	Pashto
Pashto	Bengali	Tok Pisin	Guaraní
Turkish	Haitian Creole	Cebuano	Igbo
Tagalog	Lao	Kazakh	Amharic
Vietnamese	Zulu	Telugu	Mongolian
	Tamil	Lithuanian	Javanese
		Swahili	Dholuo
			Georgian

N.B. These will be available from the LDC at $US 25.00 per language for non-members.

The IARPA Babel Program

"...to rapidly develop speech recognition capability for keyword search in a previously unstudied language, working with speech recorded in a variety of conditions with limited amounts of transcription."

Limited resources

Hours of transcribed training data

Period	Hours
1	100
2	10
3	3
4	40

N.B. In Periods 3 and 4, no phonetic lexicons.

The IARPA Babel Program

"...to rapidly develop speech recognition capability for keyword search in a previously unstudied language, working with speech recorded in a variety of conditions with limited amounts of transcription."

What is keyword search, and why focus on it?

Detection task: given

- a word or short phrase and
- a collection of speech data,

does it occur, and if so where does it occur, and how confident are you?

We can build practical keyword search from unreliable speech recognition.

Keyword search topics

- Weighted finite-state acceptors and transducers
- Constructing an audio index
- Constructing queries
- Measuring keyword search performance

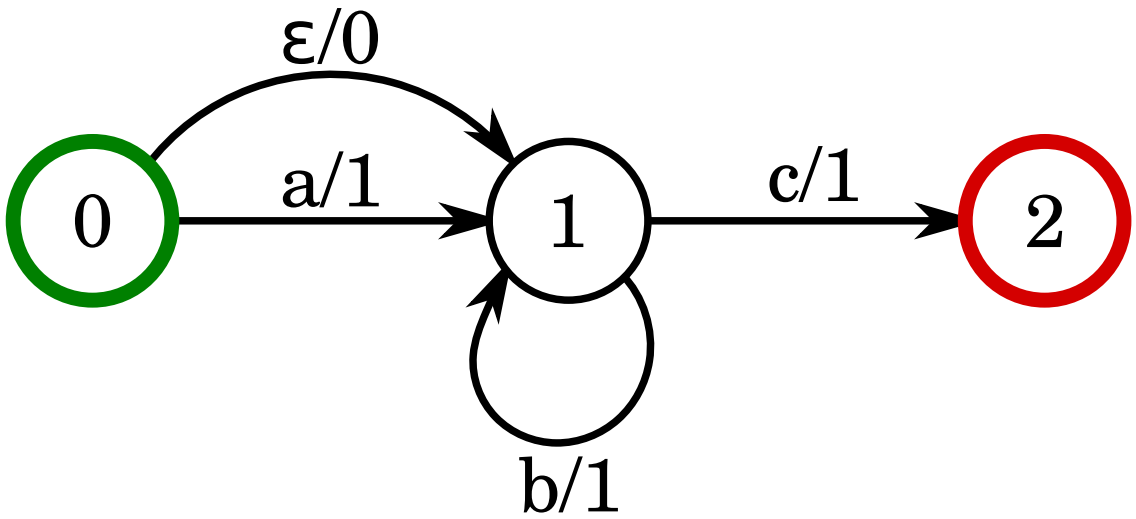
Weighted Finite-State Acceptors

A weighted finite-state acceptor compactly represents a set of strings, with a score assigned to each string.

Formally, a WFSA comprises

- states some of which are start states, end states, or both; and
- edges between states, labeled with symbols from a finite alphabet and scores.

An Example



Represents `a?b*c` and the score of each string is its length.

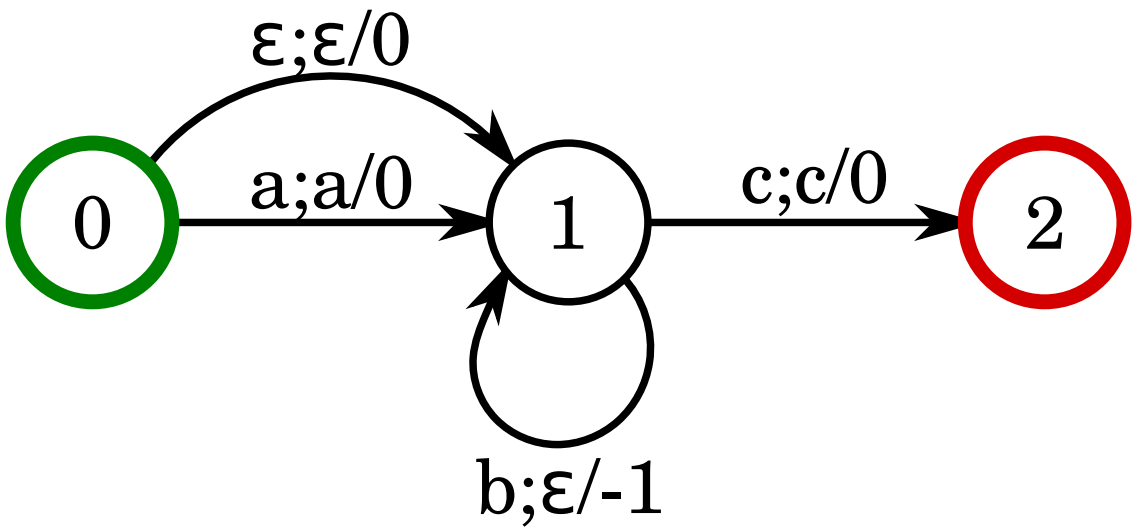
Weighted Finite-State Transducers

A weighted finite-state transducer compactly represents a relation between two sets of strings, with a score assigned to each output string.

Formally, a WFST comprises

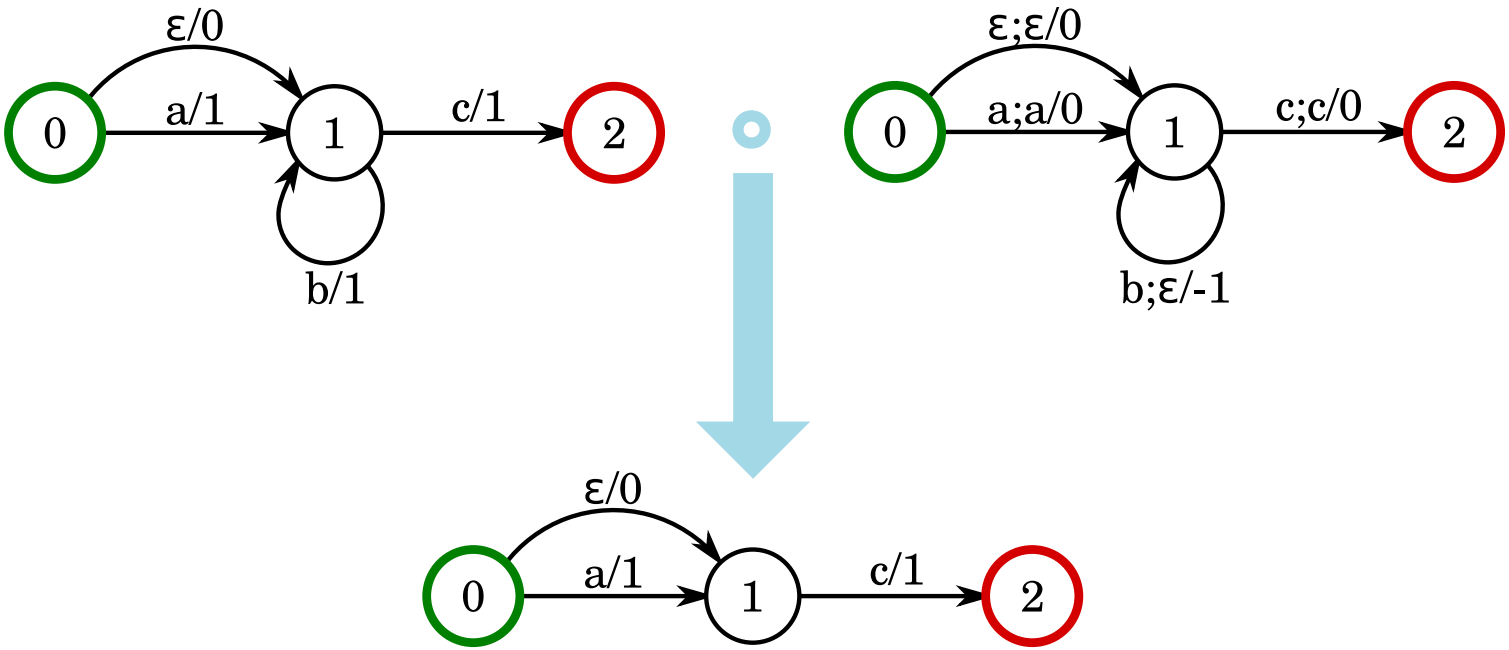
- **states** some of which are start states, end states, or both; and
- **edges** between states, labeled with input symbols from a finite alphabet, output samples from a (potentially different) finite alphabet, and scores.

An Example



Maps strings in `a?b*c` to `a?c` by deleting `b` symbols and the score of each string the change in its length.

Composition



Given WFSA $\mathcal{S}$ and WFST $\mathcal{T}$, their composition $\mathcal{T} \circ \mathcal{S}$ is a WFSA that represents the set of strings (and corresponding scores) obtained by applying $\mathcal{T}$ to the set of strings represented by $\mathcal{S}$.

Roadmap for WFST-based keyword search

1. Form a WFST index representing a relation between

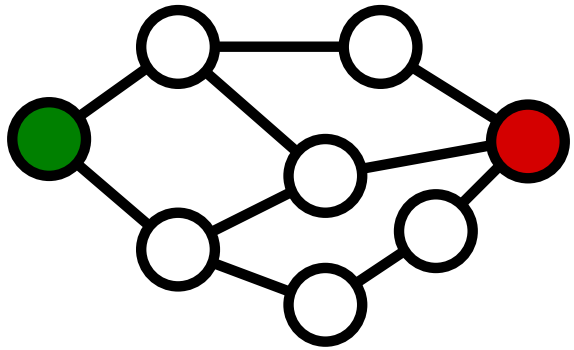
   inputs all high-probability strings of words or phones in the collection,
   outputs times of occurrence of the words or phones, and
   scores probabilities of the strings;

2. Form a WFSA representing a query; then

3. compose the query WFSA with the index WFST to perform the search.

Building an index

1. Generate a lattice for each segment in the collection.

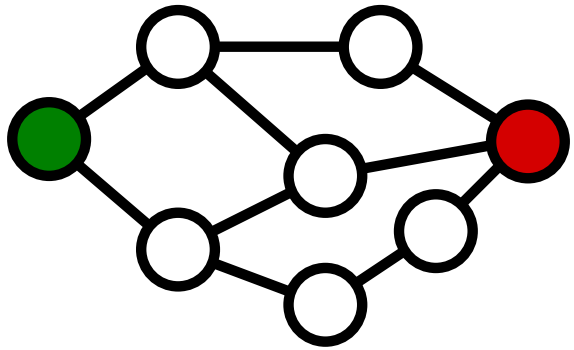


Lattice

Nodes	times
Edges	words (or phones) and posterior probabilities

Building an index

1. Generate a lattice for each segment in the collection.

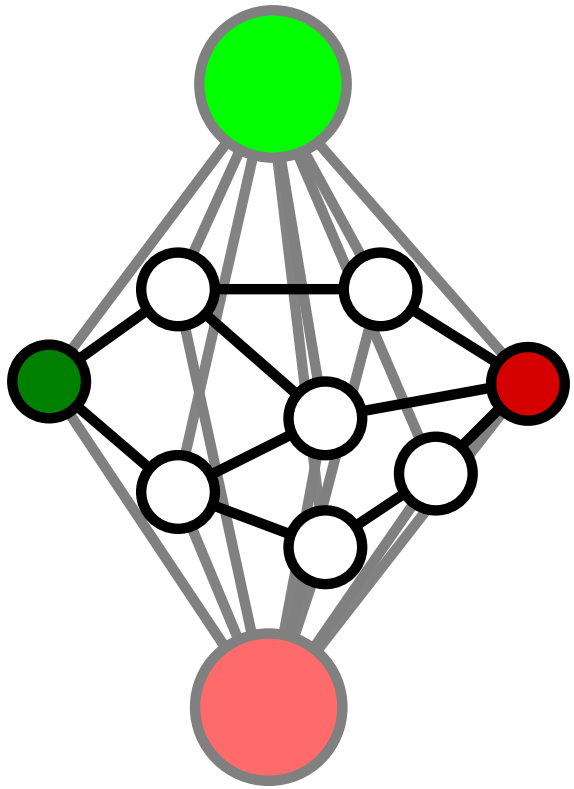


Transducer

Inputs	words (or phones)
Outputs	times
Scores	negative log-posteriors

Building an index

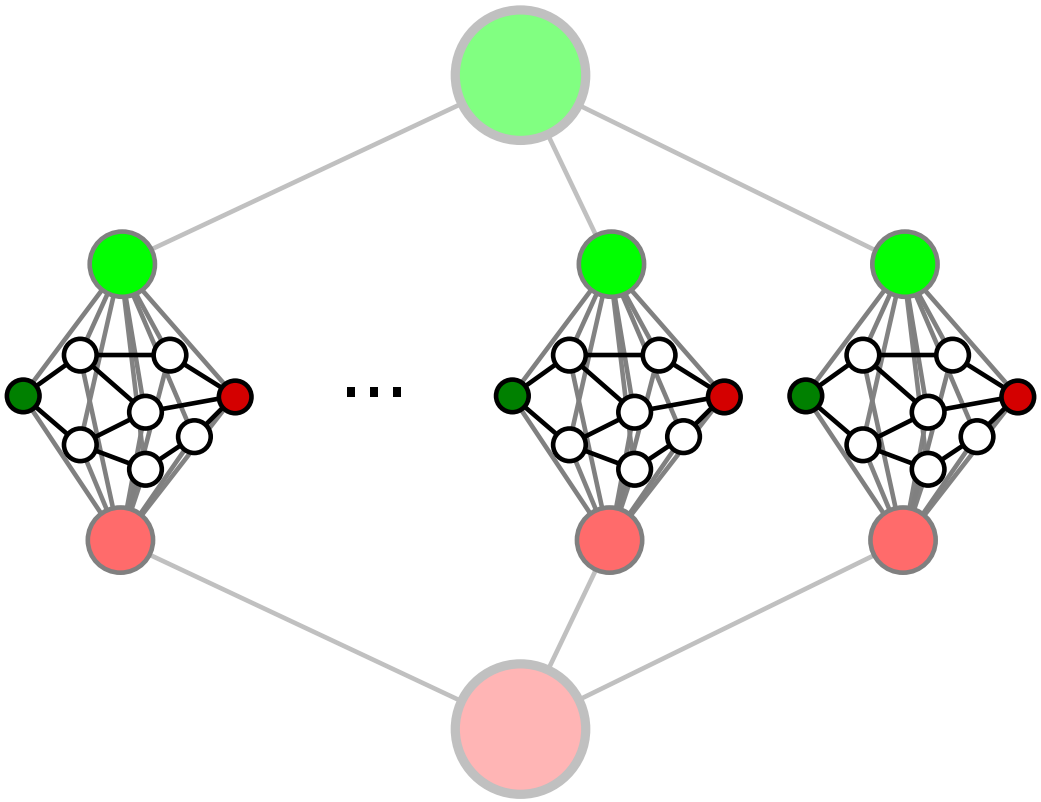
2. Produce the factor automaton for each segment.



Added edges have ϵ inputs, ϵ outputs, and no costs.

Building an index

3. Connect all the factor automata in parallel.



Added edges from start have ϵ inputs, segment ID outputs, and no costs. Added edges to end have ϵ inputs, ϵ outputs, and no costs.

Building a query

In-vocabulary queries are simple:

1. use a word-based WFST index, and
2. represent queries as word-level linear-chain WFSAs.

Queries containing out-of-vocabulary words are more challenging.

Handling out-of-vocabulary queries

1. represent queries as phone-level linear-chain WFSAs,
2. compose the WFSAs with a WFST confusability model,
3. prune the resulting WFSAs, and
4. **if a word-based index is used**, compose the query WFSAs with a WFST that maps phone sequences back to words.

The confusability model is created by running speech recognition on the training data and counting co-occurrences of reference and hypothesized phones.

N.B. This is a form of query expansion, which is well studied in the information retrieval literature.

How do we measure keyword search performance?

Actual Term-Weighted Value (ATWV)

$$\mathrm{ATWV}(\theta) = \frac{1}{N} \sum_{q} \left[\frac{N_{\mathrm{hit}}(q;\theta)}{N_{\mathrm{true}}(q)} - \beta \frac{N_{\mathrm{FA}}(q;\theta)}{T - N_{\mathrm{true}}(q)} \right]$$

- There are N test queries indexed by q,
- θ is a decision threshold,
- $N_{\mathrm{hit}}(q;\theta)$ is the number of correct detections and
- $N_{\mathrm{FA}}(q;\theta)$ is the number of false alarms for query q at decision threshold θ,
- $N_{\mathrm{true}}(q)$ is the number of occurrences of query q in the test audio,
- T is the duration of the test audio in seconds, and
- $\beta = 999.9$ is a weight on the penalty for false alarms.

Breaking it down

$$\text{ATWV}(\theta) = \frac{1}{N} \sum_{q} \left[\frac{N_{\text{hit}}(q;\theta)}{N_{\text{true}}(q)} - \beta \frac{N_{\text{FA}}(q;\theta)}{T - N_{\text{true}}(q)} \right]$$

- Hits earn a reward of $\frac{1}{N \cdot N_{\text{true}}(q)}$, so rare terms are more valuable.

N.B. Score normalization is important under the ATWV metric. For more details on this issue, please see [WM14].

Breaking it down

$$\text{ATWV}(\theta) = \frac{1}{N} \sum_{q} \left[\frac{N_{\text{hit}}(q;\theta)}{N_{\text{true}}(q)} - \beta \frac{N_{\text{FA}}(q;\theta)}{T - N_{\text{true}}(q)} \right]$$

- Hits earn a reward of $\frac{1}{N \cdot N_{\text{true}}(q)}$, so rare terms are more valuable.
- False alarms incur a constant loss of approximately $\frac{\beta}{N \cdot T}$ (because $T \gg N_{\text{true}}(q)$ for all q).

N.B. Score normalization is important under the ATWV metric. For more details on this issue, please see [WM14].

Breaking it down

$$\mathrm{ATWV}(\theta) = \frac{1}{N} \sum_{q} \left[\frac{N_{\mathrm{hit}}(q;\theta)}{N_{\mathrm{true}}(q)} - \beta \frac{N_{\mathrm{FA}}(q;\theta)}{T - N_{\mathrm{true}}(q)} \right]$$

- Hits earn a reward of $\frac{1}{N \cdot N_{\mathrm{true}}(q)}$, so rare terms are more valuable.
- False alarms incur a constant loss of approximately $\frac{\beta}{N \cdot T}$ (because $T \gg N_{\mathrm{true}}(q)$ for all q).
- Perfect performance scores ATWV $= 1.0$, while empty output scores ATWV $= 0.0$. ATWV < 0.0 means you have too many false alarms!

N.B. Score normalization is important under the ATWV metric. For more details on this issue, please see [WM14].

Babel Languages

Period 1	Period 2	Period 3	Period 4
Cantonese	Assamese	Kurmanji Kurdish	Pashto
Pashto	Bengali	Tok Pisin	Guaraní
Turkish	Haitian Creole	Cebuano	Igbo
Tagalog	Lao	Kazakh	Amharic
Vietnamese	Zulu	Telugu	Mongolian
	Tamil	Lithuanian	Javanese
		Swahili	Dholuo
			Georgian

What language characteristics matter?

1. Morphology
2. Writing system
3. Tonal languages

Morphology

The morphology of a language refers to the structure of its words. Specifically, are words atomic, or are they composed from meaningful parts?

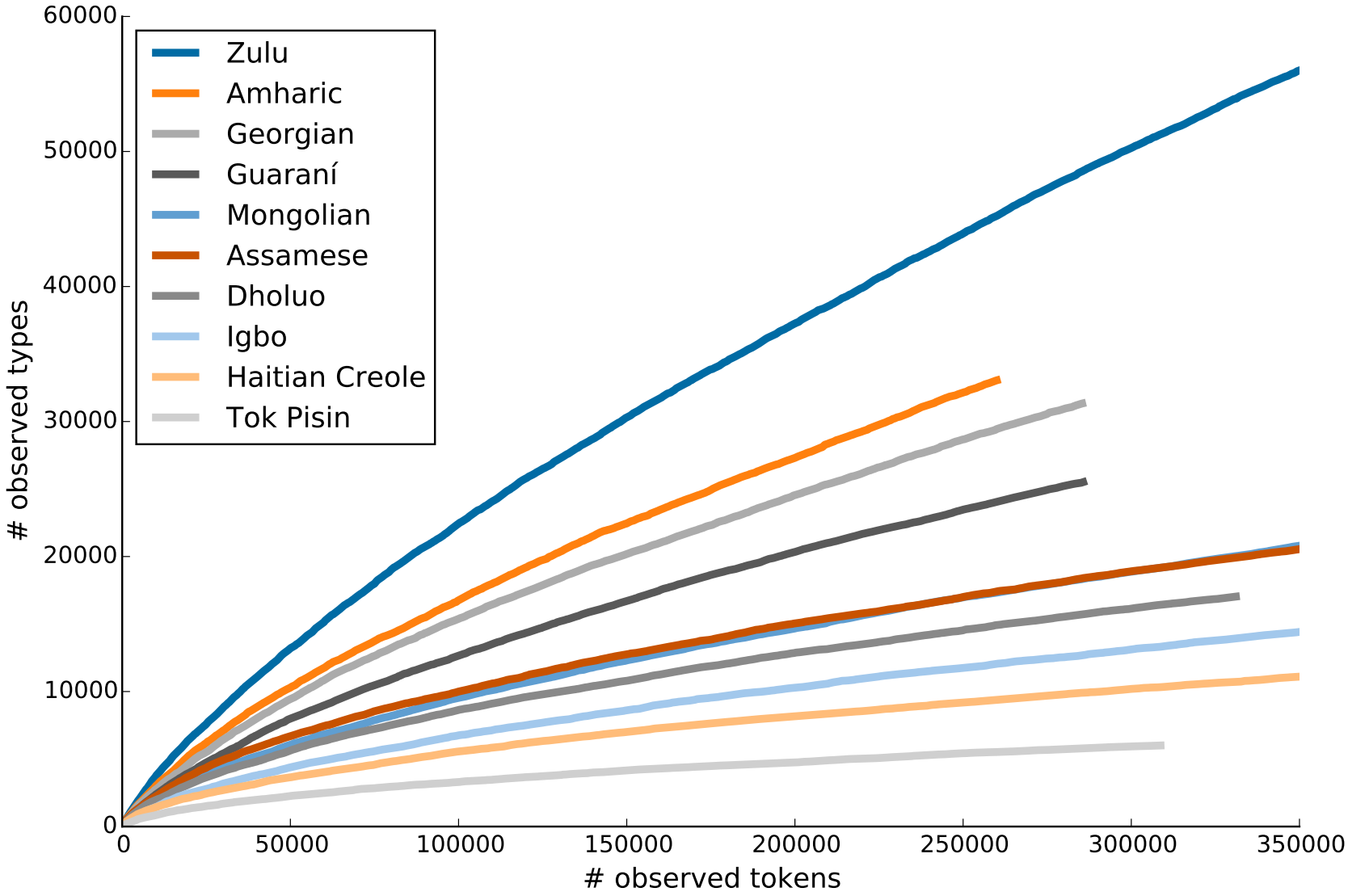
English example

re+ +do+ +er — one who is performing some action again

Languages with agglutinative morphology have large inventories of words formed by the composition of smaller units (morphs).

One way to characterize this is through vocabulary growth.

Vocabulary Growth for Babel Languages



Mitigating Rapid Vocabulary Growth

Out-of-vocabulary words are more likely to occur in test data for languages with rapid vocabulary growth, which will degrade speech recognition and keyword search performance.

Two ways to reduce the impact:

morph models base the vocabulary on morphs instead of words, and

web text expand the vocabulary by collecting text from the web.

Morph Models

Recipe:

1. Segment the training text into morph-like units using Morfessor [VSGK13] or some other tool.
2. Generate a pronunciation lexicon for the morphs.
3. Train a language model on the segmented training text.

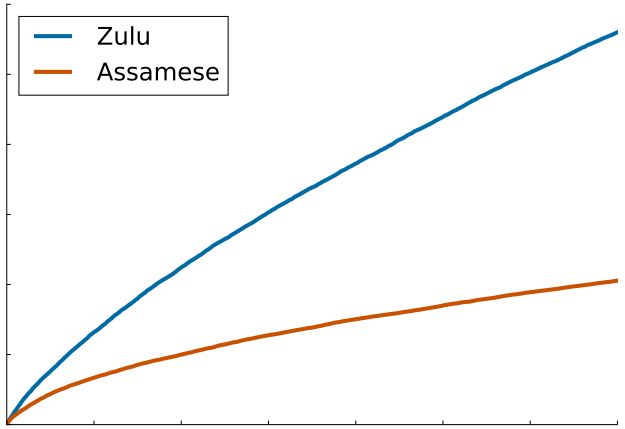
Task-specific considerations:

ASR
- Recomposing morphs into words is challenging.

KWS
- Need to decompose queries into morphs.
- Best results obtained by doing word and morph search, and fusing the results.

Keyword Search Results with Morph Models

Language	Word MTWV	Word+Morph MTWV
Zulu	0.1817	0.2305
Assamese	0.2643	0.2797



N.B. Maximum Term-Weighted Value (MTWV) is ATWV with an oracle decision threshold.

Web Text

General considerations:

- How much web presence does the language have?
  - Swahili: plenty
  - Dholuo: sparse
- How easy is it to generate queries returning the target language?
  - Amharic has its own Unicode range, so it is easy.
  - Tagalog contains many English and Spanish loanwords, so it is difficult.
- Normalization is a must!

Web Text

Task-specific considerations:

- **ASR** Improvements may be small due to genre mismatch (LM is important).
- **KWS** Better vocabulary coverage improves performance (LM is not important).

Impact of Web Text

Language	Web?	Train size	Voc. size	% WER	MTWV
Igbo	N	45.3	16.5	66.5	0.2387
	Y	17200.	19.5	66.5	0.2389
Guaraní	N	46.8	26.3	53.5	0.4562
	Y	23900.	36.9	53.5	0.4665
Mongolian	N	51.2	23.6	61.1	0.3497
	Y	118000.	220.	60.4	0.3849

Word counts are in thousands.

How the Writing System Affects Lexicon Design

We will discuss four types of writing system:

logographic Large character inventory, with each character representing a morpheme, e.g. Chinese.

alphabetic Small character inventory, with characters representing phonemes, e.g. Javanese.

abjad Small character inventory, with characters representing consonants and long vowels, e.g. Pashto.

abugida Medium character inventory, with characters representing consonant-vowel sequences, e.g. Amharic.

Lexicons for Logographic Writing Systems

Unless you have thousands of hours of transcribed data, you must have a phonetic dictionary of characters.

Lexicons for Alphabets

Graphemic approach

The pronunciation of a word is its letter sequence. Use if

- you have no other information about the relationship between spelling and sound, or
- the writing system is phonetic, as in Turkish or Georgian.

Approximate phonetic approach

Use WFST to convert letter sequences to sound sequences.

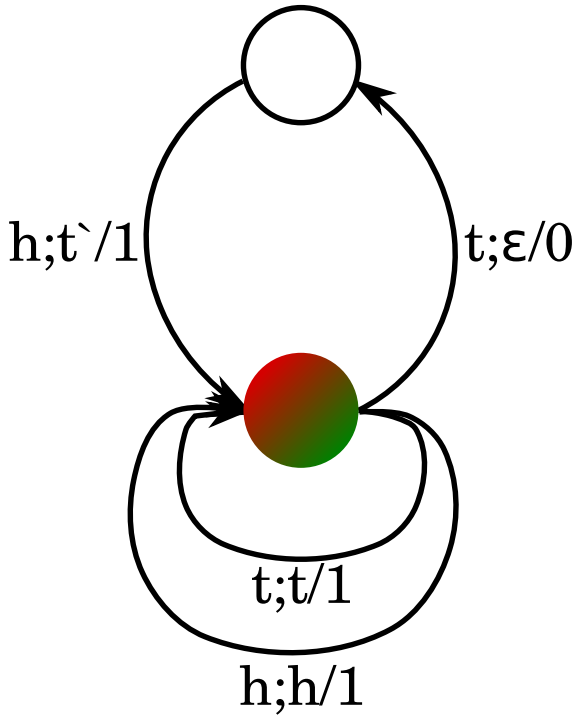
Javanese Spelling-to-sound

Javanese uses the Latin alphabet. Most letters have a single pronunciation (phonetic writing), but there are digraphs that correspond to single phonemes:

th $\rightarrow$t‘
dh$\rightarrow$d‘
ny$\rightarrow$J
ng$\rightarrow$N
sy $\rightarrow$S

Use a greedy, WFST-based pronunciation generator.

Javanese Spelling-to-sound



Compose each word with transducer and pick the output with the lowest score (length).

N.B. The diagram is a sketch that omits many transitions.

Lexicons for Abjads

Use the same approach as for alphabets, but what about the unwritten vowels?

Ignore them Simple, and usually works fine.

Infer them Feasible only with lots of transcribed data.

Lexicons for Abugidas

In general, each character will map to a consonant-vowel pair.
The details will be language-dependent.
In Amharic each Unicode code point denotes a syllable:

ሙ → m @
ሙ፡ → m u
ሚ → m i
ማ → m a
ሜ → m e

In such cases, a lookup table is enough.

Lexicons for Abugidas

In many Indian languages, such as Telugu, characters have a default vowel that can be modified by the following character.

మ → m @ (only if next character is a consonant)

ము → m u

మి → m i

మా → m a

మె → m e

In these cases, a WFST framework is the best approach.

N.B. In the list above, the first Telugu glyph is generated by one Unicode code point, but the others are generated by two.

Tonal Languages

Tonal languages use pitch (and phonation in some cases) to convey lexical or grammatical information. Tone information might or might not be indicated in the writing system:

Written	Cantonese, Vietnamese, Lao
Not written	Zulu, Igbo, Dholuo

Tonal Languages

If tone is indicated in the writing system, the front end needs to represent pitch either

- explicitly (e.g., using FFV features [LHE08]), or
- implicitly (e.g., using high-resolution Mel spectra).

Features	**% WER**
PLP	63.3
+FFV	60.6

Results on Lao, using a 10-hour training set.

A Recipe for a New Language: Paraguayan Guaraní

- Background on the language
- Dictionary generation
- Training acoustic models
- Training language models
- Performance comparisons

Paraguayan Guaraní

- An official language in Paraguay
- Tupian language family
- Around 4 million native speakers
- Substantial Spanish influence: code-switching and calques

N.B. Map of Paraguay by Wikimedia user Rei-artur

Dictionary Generation

The Guaraní writing system

- Latin alphabet
- 'w' only in loan words
- apostrophe marks glottal stops
- Tilde can appear on eight letters
  - ã, ẽ, ĩ, õ, ũ, ỹ, g̃, and ñ
- Acute accent can appear on six letters
  - á, é, í, ó, ú, ý
- Dieresis can appear on 'u'
  - ü
- Ten digraphs
  - mb, nd, ng, and nt (prenasalized consonants)
  - g̃ (Unicode Latin Small Letter G and Unicode Combining Tilde)
  - ch, sh, and qu
  - rr and ll (only in Spanish loan words)

Sample Lexicon Entries

Regular words

emba'apomiéma(01)	e mb a ' a p o m i e' m a
hasypáma(01)	h a s y p a' m a
mbói(01)	mb o' i
nopenamo'ãi(01)	n o p e n a m o ' a~ i
ofalla'imi(01)	o f a ll a ' i m i
oñakãkarãi(01)	o n~ a k a~ k a r a~ i
pack(01)	p a k
rohechaga'ueterei(01)	r o h e ch a g a ' u e t e r e i
veintinuéve(01)	v e i nt i n u e' v e
ñañomongeta(01)	n~ a n~ o m o ng e t a

Pronunciations of Spellings

The data includes spellings, which are marked with underscores, e.g. “a_m.”

In Guaraní, spellings are usually given with Spanish letter pronunciations, except for nasalized vowels and g̃.

We opted to generate both Spanish and Guaraní pronunciations for spelled tokens.

Sample Lexicon Entries

Spellings

a_m(01)	' a' ' e' m e~
a_m(02)	' a' m e~
b_c(01)	b e' s e'
b_r(01)	b e' ' e' r e
b_r(02)	b e' r e'
c_n_c(01)	s e' ' e' n e~ s e'
c_n_c(02)	s e' n e~ s e'
e_p_p(01)	' e' p e' p e'
j_r_h(01)	h o' t a ' e' r e ' a' ch e
j_r_h(02)	j e' r e' h e'

Training Acoustic Models

- Flat start: bootstrapping the models
- Monolingual deep neural network model
- Multilingual deep neural network model

Flat Start

1. Initialize single-Gaussian models with uniform segmentation
2. Run many iterations of forward-backward training
3. Gradually increase the number of mixture components

Flat Start

1. Initialize single-Gaussian models with uniform segmentation
2. Run many iterations of forward-backward training
3. Gradually increase the number of mixture components

Two pitfalls:

- Need for a garbage (REJ) model
- Inclusion of long silences within segments

Our Recipe

GMM 80 passes over data, splitting mixtures every 8.

Speech/Nonspeech

1. Seed SPEECH, SIL, and REJ models with GMM.
2. 5 iterations Viterbi training, skipping utterances containing REJ.

Context-Independent

1. Seed phone models with SPEECH.
2. 5 iterations forward-backward training, skipping utterances containing REJ.

These models use PLP+$\Delta + \Delta^2$ features.

Our Recipe

Phonetic mixup Train 3 successive models

1. 250 states; 6 mixtures per state
2. 500 states; 6 mixtures per state
3. 1,000 states; 6 mixtures per state

Training is 20 iterations of Viterbi, then 10 iterations of forward-backward.

Global mixup Train 3 successive models, each with 1,000 states and 6 mixtures per state.

Full model 6,000 states and 6 mixtures per state

These models use PLP+LDA+STC features.
Phonetic models use phone+position roots for decision trees, while global and full models use position roots.

Our Recipe

Monolingual DNN

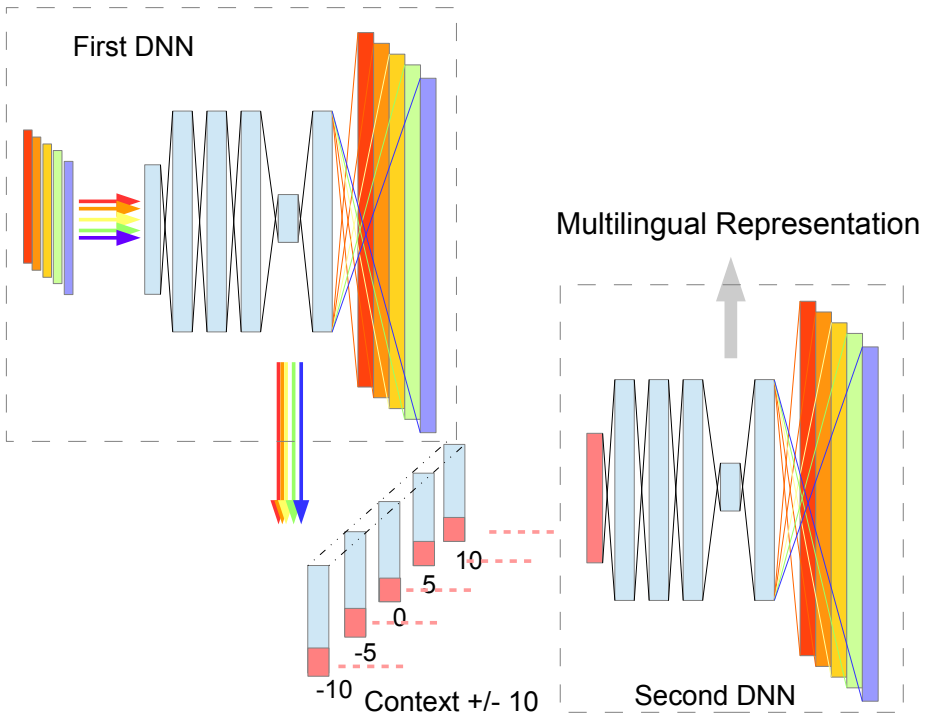
- Input is 9 frames of 40-d PLP+LDA+STC features.
- 3 hidden layers with 1,024 ReLU units, then
- 1 hidden layer with 1,024 sigmoid units, then
- 128-unit linear bottleneck [SKS+13], then
- 3,000-unit softmax output layer.

Training on 40h of data in three phases:

1. discriminative pretraining [SLCY11],
2. cross-entropy training, and
3. state-level minimum Bayes risk training with distributed Hessian-free algorithm [KSS12].

Our Recipe

Multilingual Features



Stacked CNN/DNN architecture similar to [GKB09], using language-dependent output layers [SLF+08].

N.B. Thanks to Jia Cui of IBM for this figure.

Our Recipe

Multilingual Features

CNN

- 11 frames of log-Mel+Δ + Δ^2 features
- Conv. layer with 9 kernels, 128 feature maps, and max-pooling in frequency with a window and stride of 3.
- Conv. layer with 3 kernels, 256 feature maps, and no pooling.
- 7 hidden layers of 2048, 1024, 1024, 1024, 80, 1024, and 3000 sigmoidal units.

DNN

- 80-d bottleneck output from CNN at $t-10$, $t-5$, t, $t+5$, and $t+10$.
- 5 hidden layers of 1024, 1024, 80, 1024, and 3000 sigmoidal units.

Training of all networks is cross-entropy on 24 Babel languages (all but Georgian).

Our Recipe

Multilingual DNN

- Input is 1 frame of 80-d multilingual features.
- 3 hidden layers with 1,024 ReLU units, then
- 1 hidden layer with 1,024 sigmoid units, then
- 256-unit linear bottleneck, then
- 3,000-unit softmax output layer.

Training on 40h of data in three phases:

1. discriminative pretraining,
2. cross-entropy training, and
3. state-level minimum Bayes risk training with distributed Hessian-free algorithm.

Training Language Models

On transcripts Bigram LM with modified Kneser-Ney smoothing trained on 320K words with a 26.3K vocabulary.

Plus web text Interpolated bigram LM with modified Kneser-Ney smoothing trained on 17.2M words with a 36.9K vocabulary.

Performance of different acoustic models

Model	Multilingual?	% WER
GMM	N	68.9
CE DNN	N	63.7
sMBR DNN	N	58.3
sMBR DNN	Y	47.6

All runs use bigram LM trained from transcripts.

Performance of different language models

Model	% WER	MTWV
Transcripts	47.6	0.5289
+ Web text	47.5	0.5345

All runs use multilingual DNN acoustic model.

Tackling a New Domain

Tackling a New Domain

Outline

1. Acoustic models for New Domains
   - With no adaptation data
     - Robust Features
     - Feature compensation and test-time adaptation
     - Multicondition training
   - With adaptation data
     - Model adaptation
2. Language models for New Domains
   - Commonly seen words in the new domain
   - In-Domain data and other related data sources - Web data
3. Recipes for New Domains

Acoustic Models in New Domains

Robustness - Things Change

- Background noise can increase or decrease.
- Channel can change.
  - Different microphone.
  - Microphone placement.
  - Location characteristics - Reverberation effect
- Speaker characteristics vary.
  - Different vocal tract lengths (gender)
  - Different speaking rates.
  - Different accents
  - Different age
  - There will always be speakers or conditions not represented well in the training data.
- Heaven knows what else can happen.

Acoustic models in ASR

Recap:Probabilistic Model for Speech Recognition

$$
\begin{aligned}
w^* &= \arg\max_{w \in vocab} P(w|x,\theta) \\
&= \arg\max_{w \in vocab} \frac{P(x|w,\theta)P(w|\theta)}{P(x)} \\
&= \arg\max_{w \in vocab} P(x|w,\theta)P(w|\theta)
\end{aligned}
$$

- w^* Best sequence of words
- x Sequence of acoustic vectors
- θ Model Parameters

Acoustic Models in New Domains

What happens when things change?

Recognition performance falls apart!
Why?

Acoustic Models in New Domains

What happens when things change?

Recognition performance falls apart!
Why?
Because the features on which the system was trained have changed.

Acoustic Models in New Domains

What happens when things change?

Recognition performance falls apart!
Why?
Because the features on which the system was trained have changed.
How do we mitigate the effects of such changes? What components of the system should we look at?

Acoustic Models in New Domains

What happens when things change?

Recognition performance falls apart!
Why?
Because the features on which the system was trained have changed.
How do we mitigate the effects of such changes? What components of the system should we look at?
The Acoustic Model: $P(x|W,\theta)$ and the features x seem to be the most logical choices. So what can we do?

Acoustic Models in New Domains

Robustness Strategies

Acoustic Models in New Domains

Robustness Strategies

- **Re-training:** Retrain system using the changed features.

Acoustic Models in New Domains

Robustness Strategies

- **Re-training:** Retrain system using the changed features.
- **Robust features:** Features x that are independent of noise, channel, speaker, etc. so θ does not have to be modified.

Acoustic Models in New Domains

Robustness Strategies

- **Re-training:** Retrain system using the changed features.
- **Robust features:** Features x that are independent of noise, channel, speaker, etc. so θ does not have to be modified.
  - More an art than a science but requires little/no data.

Acoustic Models in New Domains

Robustness Strategies

- **Re-training:** Retrain system using the changed features.
- **Robust features:** Features x that are independent of noise, channel, speaker, etc. so θ does not have to be modified.
  - More an art than a science but requires little/no data.
  - Even with state-of-the-art neural networks that take the input audio as input to the first layer
- **Modeling:** Explicit models for the effect the distortion (Noise, channel,speaker) has on speech recognition parameters $\theta' = f(\theta, D)$.

Acoustic Models in New Domains

Robustness Strategies

- **Re-training:** Retrain system using the changed features.
- **Robust features:** Features x that are independent of noise, channel, speaker, etc. so θ does not have to be modified.
  - More an art than a science but requires little/no data.
  - Even with state-of-the-art neural networks that take the input audio as input to the first layer
- **Modeling:** Explicit models for the effect the distortion (Noise, channel,speaker) has on speech recognition parameters $\theta' = f(\theta, D)$.
  - Works well when model fits, requires less data.

Acoustic Models in New Domains

Robustness Strategies

- **Re-training:** Retrain system using the changed features.
- **Robust features:** Features x that are independent of noise, channel, speaker, etc. so θ does not have to be modified.
  - More an art than a science but requires little/no data.
  - Even with state-of-the-art neural networks that take the input audio as input to the first layer
- **Modeling:** Explicit models for the effect the distortion (Noise, channel,speaker) has on speech recognition parameters $\theta' = f(\theta, D)$.
  - Works well when model fits, requires less data.
- **Adaptation:** Update estimate of θ from new observations.

Acoustic Models in New Domains

Robustness Strategies

- **Re-training:** Retrain system using the changed features.
- **Robust features:** Features x that are independent of noise, channel, speaker, etc. so θ does not have to be modified.
  - More an art than a science but requires little/no data.
  - Even with state-of-the-art neural networks that take the input audio as input to the first layer
- **Modeling:** Explicit models for the effect the distortion (Noise, channel,speaker) has on speech recognition parameters $\theta' = f(\theta, D)$.
  - Works well when model fits, requires less data.
- **Adaptation:** Update estimate of θ from new observations.
  - Very powerful but often requires the most data $\theta' = f(D, p(O|W, \theta))$.
  - Goal is to have an algorithm that adapts a large number of parameters with as little data from the new domain as possible

General Retraining

If the environment changes, retrain system from scratch in new environment.

- Very expensive - cannot collect hundreds of hours of data for each new environment.
- Also very expensive computationally and time-wise

Two strategies.

- Environment simulation.
- Multistyle Training.

Environment Simulation

- Take training data
- Measure parameters of new environment
- Transform training data to match new environment
- Retrain system, hope for the best

Multistyle Training

- Take training data.
- Corrupt/transform training data in various representative fashions (data augmentation)
- Collect training data in a variety of representative environments.
- Pool all such data together; retrain system.

Issues with Retraining

Simplistic models of degradations

- E.g. telephony degradations more than just a different compression scheme based on the codec used

Hard to anticipate every possibility.

- In high noise environment, person speaks louder with resultant effects on glottal waveform, speed, etc.

Therefore other schemes - noise modeling and general forms of adaptation - are needed and sometimes used in tandem with these other schemes.

Robust Features: Handling noise and channel variability

Cepstral Mean Normalization

Basic Idea: Assume all speech is linearly filtered but that the linear filter changes slowly with respect to speech (many seconds or minutes).

Effects from the channel or environment are usually modeled as additive or convolutive distortions

Given a set of cepstral vectors O_t we can compute the mean:

$$\bar{O} = \frac{1}{N} \sum_{t=1}^{N} O_t$$

Robust Features (contd.)

Linear channels have a convolutive effect - $y[n] = x[n] * h[n]$, with filter responses $h[n]$ that have a small time constants (less than 25ms).
For frame index k

$$y_k[n] = x_k[n] * h[n]$$

Short-term cepstra (DCT of power spectrum) is modified by the linear channel,

$$Y_k(\omega) = X_k(\omega)H(\omega)$$
$$\log(|Y_k(\omega)|^2) = \log(|X_k(\omega)|^2) + log(|H(\omega)|^2)$$
$$C_y[k] = C_x[k] + C_h$$

Robust Features (contd.)

- To mitigate the effect of the linear channel , In "Cepstral mean normalization" we subtract the mean of a set of cepstral vectors from each vector individually

$$\hat{O}_t = O_t - \bar{O}$$

- When the speech signal is distorted with additive noise, $y[n] = x[n] + q[n]$, uncorrelated with the speech signal, its effect can be mitigated by subtracting the estimate of the noise from the signal. Using a voice activity detector, the noise power spectral estimate is obtained as mean of the power spectrum in the non-speech region and subtracted.

$$|\hat{X}_k(\omega)|^2 = |Y_k(\omega)|^2 - |\hat{Q}(\omega)|^2$$

Robust Features (contd.)

Reverberant speech is also modeled as a long-term convolution effect:

$$Y_k(\omega) = X_k(\omega)R(\omega)$$
$$\log(|Y_k(\omega)|) = \log(|X_k(\omega)|) + log(|R(\omega)|)$$

Its effect is mitigated by subtracting the long-term mean of $\log(|Y_k(\omega)|)$

Issues

- Must be performed on both training and test data.
- Bad things happen if utterances are very short
- Bad things happen if there is a lot of variable length silence in the utterance

Types of Acoustic Model Adaptation

- Supervised vs. Unsupervised: Is correct transcription of utterance available at test time ?
- Batch vs.Incremental adaptation: Whole vs. small (time critical) portion of the test data is available (ideally suited for real-time systems)
- What do we do if correct transcript is not available ?
  - We obtain a first pass hypothesis using the best ASR system.
  - Use the above recognition result instead of the correct transcript.

Handling speaker variability

The goal is to fit acoustic model or features to the target speaker using the new acoustic adaptation data from the target speaker

- Model-based - Feature-based:
  Adaptation is the process of modifying the model to better fit the features (eg. Maximum Likelihood Linear Regression)
  Normalization is the process of transforming features to better fit the model (eg. cepstral mean subtraction, feature based Maximum Likelihood Linear Regression)
  - Can be done in a supervised or unsupervised manner
  - Can be implemented in batch or incremental mode

But, we have a continuous stream of audio!

Where are the speakers?

Properties: Speaker identity, recording condition (e.g. background noise, telephone channel), signal type (e.g. speech. music, noise, silence), and spoken content.
Segmentation and Clustering

- Segmentation: Partitioning of the audio into homogeneous areas, ideally we should have one speaker/condition per segment.
- Clustering: Locating speakers and conditions involves the clustering of these segments into similar speakers/conditions.

Processing the audio

The performance of an ASR system is affected by:

- Segmentation affects speech recognition performance: speaker adaptation and speaker clustering assume one speaker per segment
- Language model assumes sentence boundaries at the end of segments
- Non-speech regions in the segment are often filled with noise and cause insertion errors
- Overlapping speech is not recognized correctly, causing errors in the surrounding regions

Segmenting the audio stream

Hypothesize segment boundaries:

- Input stream is modeled as Gaussian process in the cepstral domain.
- Feature vectors of one segment: drawn from multivariate Gaussian: $O_i^j := O_i \dots O_j \sim \mathcal{N}(\mu, \Sigma)$
- For hypothesized segment boundary t in O_1^T decide between
  - $O_1^T \in \mathcal{N}(\mu, \Sigma)$ and
  - $O_1^t \in \mathcal{N}(\mu_1, \Sigma_1)$ $O_{t+1}^T \in \mathcal{N}(\mu_2, \Sigma_2)$
- Use difference of Bayesian Information Criterion values:

$$\Delta\mathrm{BIC}(t) = \mathrm{BIC}(\mu, \Sigma, O_1^T) - \mathrm{BIC}(\mu_1, \Sigma_1, O_1^t) - \mathrm{BIC}(\mu_2, \Sigma_2, O_{t+1}^T)$$

Clustering of segmented audio

- Group speech segments into clusters for adaptation.
- Segments from same or similar speakers and conditions should be grouped together
- Greedy, bottom up clustering using acoustic features that merges segments
- Use Bayesian Information Criterion to control number of clusters that yields the best performance of the ASR system using one of the adaptation schemes mentioned in the next few slides

Two classic Model Adaptation schemes for GMMs

Maximum A Posteriori - Basic Idea

One way to achieve robustness is to take a fully trained HMM system, a small amount of data from a new domain, and combine the information from the old and new systems together.

In Maximum A Posterior Estimation we assume there is some prior probability distribution on θ, $p(\theta)$ (the ASR system) and we try to pick $\hat{\theta}$ to maximize the a posteriori probability of θ given the observations (domain specific data):

$$\begin{aligned} \hat{\theta} &= \arg\max_{\theta} \; p(\theta|O_1^N) \\ &= \arg\max_{\theta} \; \mathcal{L}(O_1^N|\theta)p(\theta) \end{aligned}$$

Maximum a Posteriori Adaptation

MAP

1. Align adaptation data against a set of existing HMM models.
2. Collect counts $C_t(i,j)$, the fractional count (aka the posterior probability) at time t for being in mixture component j of state i
3. Use this to estimate parameters of the adapted model

For example, μ_{ik} is the prior estimate for the mean of mixture component k for state i from a previously trained HMM system. The adapted $\hat{\mu}_{ik}$:

$$\hat{\mu}_{ik} = \frac{\tau_{ik}\mu_{ik} + \sum_{t=1}^{N} C_t(i,k)\%\%_t}{\tau_{ik} + \sum_t C_t(i,k)}$$

τ is a parameter that can be tuned to optimize performance on different test domains.

Maximum Likelihood Linear Regression - Basic Idea

MLLR

An alternate strategy is the popular MLLR adaptation scheme.
In MAP, the different HMM Gaussians are free to move in any direction.
In Maximum Likelihood Linear Regression the means of the Gaussians are constrained to only move according to an affine transformation $Ax + b$.

Feature transforms based on adaptation data

Constrained/Feature MLLR

1. Adaptation technique used in ASR to reduce the mismatch between acoustic features from a speaker and trained models

$$\hat{\mu} = A_c \mu - b_c$$
$$\hat{\Sigma} = A_c \Sigma {A_c}^T$$

2. Equivalent to transforming the features

$$\hat{o_t} = {A_c}^{-1} o_t + {A_c}^{-1} b_c$$

3. Transformation parameters are estimated with EM to maximize the likelihood of the adaptation data

Acoustic Model Adaptation with Neural Networks

With the success of Neural Networks as a replacement for GMMs, all state-of-the-art ASR systems use a variant of deep neural networks.
Do the adaptation strategies for new domains still hold with neural networks?

Acoustic Model Adaptation with Neural Networks

With the success of Neural Networks as a replacement for GMMs, all state-of-the-art ASR systems use a variant of deep neural networks.
Do the adaptation strategies for new domains still hold with neural networks?
Stay tuned . . .

Acoustic Model Adaptation with Neural Networks

Adaptation of neural network based acoustic models - involves adapting a large number of parameters with a small amount of adaptation data. Recent advances include:

- Fine-tuning - additional epochs of training of some or all layers of the network with the adaptation data alone (similar to new languages)
- Addition of additional layer to perform a feature-space-like adaptation
- Use of a regularizer to control the extent by which the weights move from the baseline model when trained with the in-domain audio
- Rely on feature adaptation (fMLLR/ivectors) to serve as medium of adaptation and subsequently become the input to the neural network

Acoustic Model Adaptation with Neural Networks

Weight decay based adaptation: Similar to MAP adaptation, adapted network's weight updates are arrived at from using a weighted combination of the updates from adaptation data and the baseline model

$$\Delta w_t = -\alpha \nabla_w E(w_t) - \beta(w_{t-1} - w_0)$$

α - learning rate, β - regularization parameter, w_0 - model parameters of the initial model

Useful for both speaker and domain adaptation

Weight decay based adaptation

Performance of adapted ASR system

Supervised Adaptation

Model	WER%
Baseline SI CNN	36.5
1 hr WD + sMBR	30.9
2 hr WD + sMBR	30.7
5 hr WD + sMBR	30.1
10 hr WD + sMBR	29.9
25 hr WD + sMBR	28.2

Unsupervised Adaptation

Model	WER%
Baseline SI CNN	36.5
2 hr WD + sMBR	32.8
25 hr WD + sMBR	31.4
75 hr WD + sMBR	30.4
200 hr WD + sMBR	30.3

How well do these techniques perform in unseen domains?

ASR in unseen domains - the ASPIRE challenge

- Construct robust automatic speech recognition systems that without having access to matched training and development data.
  - Training set - the Fisher conversational telephone training corpus (2,000 hours of transcribed telephone speech).
  - Test set - Microphone speech recorded in noisy reverberant environments.
- Techniques used by various sites:
  - Data augmentation - Add up to 3000 hours of artificially created noisy reverberant speech
  - Neural network based speech enhancement - Train a denoising autoencoder to denoise speech signals
  - Robust Features and Feature adaptation - Robust features that compensate for artifacts
  - Robust acoustic models - models that use large context
  - Multiple systems and system combination - improvements of up to 15% relative.

ASR on unseen domains - the Aurora-4 challenge

1. Aurora4 - A Medium vocabulary task, based on the Wall Street Journal corpus
   - 1.1 Data collected from a primary Sennheiser microphone, and 18 other secondary microphones.
   - 1.2 Six different noise types - airport, babble, car, restaurant, street traffic and train station, at 10-20 db SNR.
2. Techniques used by various sites:
   - 2.1 Multistyle Training
   - 2.2 Denoising autoencoder to denoise speech signals
   - 2.3 Denoising autoencoder together with noise estimates for input layer.
   - 2.4 Joint training Neural network based frontend and backend acoustic models.
   - 2.5 Varying network topologies, such as the use of a linear activation layer to capture the dynamics of acoustic features for backend NN.
3. State-of-the-art systems yield gains of up to 20% relative on unseen microphone and noise conditions.

Language models for New Domains

Does one size fit all?

Language models for New Domains

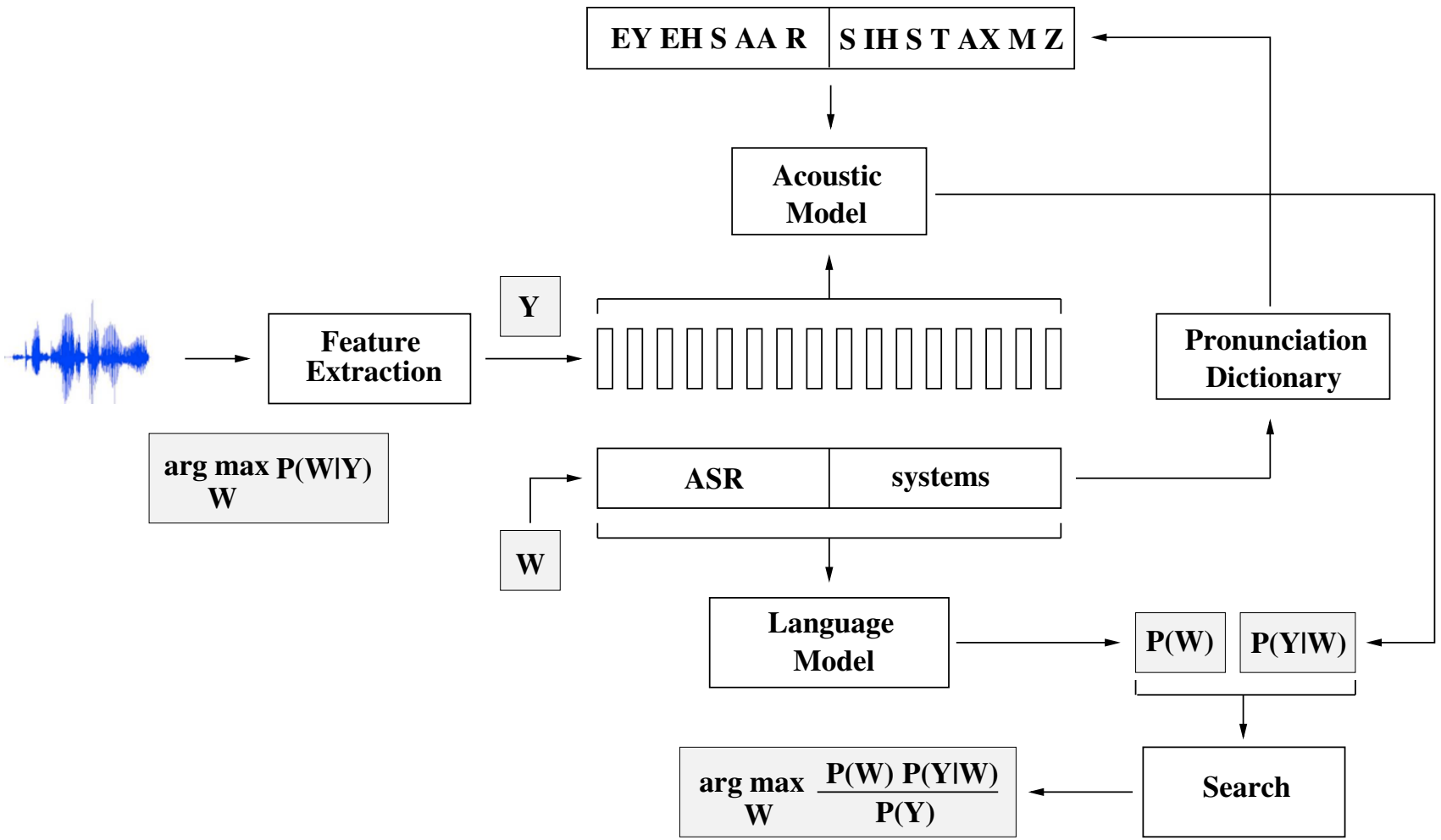
Does one size fit all?

ASR systems are required to address a broad set of applications:

- Conversation: What is the best bar at this hotel? What time is check out?
- Chit Chat: What are you doing? Are you a smart robot?
- Question Answering:What is the prognosis for Wernicke-Korsakoff Syndrome? What does Spondylolisthesis cause?
- Analytics: Call center customer support analysis for upselling, virtual agent, emotion analysis, etc.
- Transcription: Broadcast news and sports captioning, podcast and YouTube transcriptions which address a wide variety of topics

Language models for New Domains

Recap:Component blocks of ASR



Language models in ASR

Recap:Probabilistic Model for Speech Recognition

$$
\begin{aligned}
w^* &= \arg\max_{w \in vocab} P(w|x,\theta) \\
&= \arg\max_{w \in vocab} \frac{P(x|w,\theta)P(w|\theta)}{P(x)} \\
&= \arg\max_{w \in vocab} P(x|w,\theta)P(w|\theta)
\end{aligned}
$$

- w^* Best sequence of words
- x Sequence of acoustic vectors
- θ Model Parameters

Language models in ASR

Interpretation

Specific to a *domain*!!!

- Describes which word sequences are *allowed*
  - Example: restricted domains like digit strings.
- Describe which word sequences are *likely*
  - Example: unrestricted domains like web search.
  - Example: *Britney Spears* vs *Brit knee spears*.
- Analogy: multiple-choice test.
  - LM restricts choices given to acoustic model.
  - The fewer choices, the better you do.

Adapting to the new domain

Strategies: New Words without context

- Add new words (unigrams only with no context) relevant to the new domain (Out of Vocabulary (OOV) words) to the base vocabulary of the ASR system
- Pronunciations for these unigrams can be added or automatically generated
  - Manual IPA pronunciation or
  - Automatically generate pronunciations from sample acoustics or
  - Automatically generate pronunciations from the graphemes/characters
  - Derive pronunciation from a sounds-like phrase (IEEE sounds like I triple E)
- Modify the pronunciation of new or existing words to suite the domain (for example, acronyms, such as, HHonors or Hilton Honors)
- Add new words as unigrams with fixed probabilities to the in-domain LM

Adapting to the new domain

Strategies: New Words with context

- Adapt the base LM using corpora descriptive of the new-domain (text files), such as medical or legal texts with unique names and acronyms in the vocabulary
- In-domain text can be obtained either
  - from existing files from the industry (product manuals, corporate FAQ websites, training manuals, etc.) or
  - by crawling the web or other data sources using relevant in-domain keywords and obtaining large quantities of text to provide context for these new words
- In-domain text can be augmented further by adding text obtained from words that are semantically similar to the in-domain keywords (for example, derived from neural word embeddings)
  - "Speech" is semantically similar to "Address", "Statement", "Remarks", "Event ", "Ceremony"

Adapting to the new domain

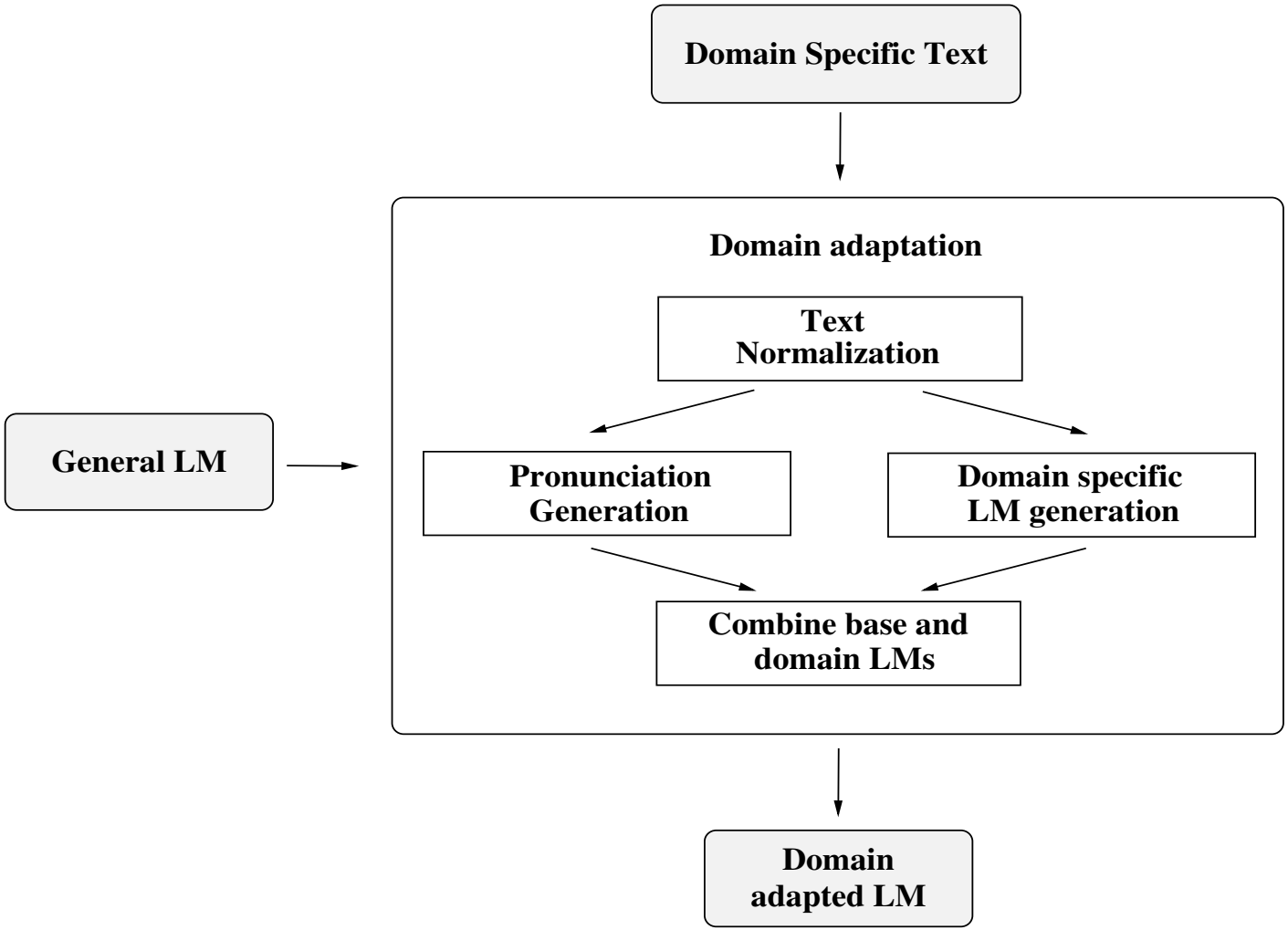
Strategies: New Words with context

- Derive new words that do not exist in the ASR system's vocabulary and add it to the lexicon
- Build an in-domain LM with the new in-domain corpora (with context for the new words)

Interpolate the in-domain LM (statically or dynamically) with the base LM for use in the ASR system to obtain the Domain Adapted LM

Domain Adaptation

Overview



Adapting to the new domain

Challenges

- Text normalization (90% of language modeling is text normalization!)
- LMs are usually very large with 100s of millions of n-grams) or complex, such as class-based exponential or neural network based models
- Can take very long to add just 10 new in-domain words and rebuild LMs!

Adapting to the new domain

Challenges

- Text normalization (90% of language modeling is text normalization!)
- LMs are usually very large with 100s of millions of n-grams) or complex, such as class-based exponential or neural network based models
- Can take very long to add just 10 new in-domain words and rebuild LMs!

Solution is ...

- Dynamically interpolate the new words and LM from the new domain
- Ensure the decoder is aware of new words in its look-ahead lists from both the base and the in-domain LMs

What problem does a Domain Adapted LM solve?

Examples

- REF: if i have had an MRSA infection or been told that i carry MRSA * am i at high risk for developing * if i get seasonal influenza
- Base LM: if i have had an IMMERSING FICTION or been told that i carry IN MARSEILLE I am i at high risk for developing IT if i get seasonal influenza
- Domain Adapted LM:: if i have had an MRSA infection or been told that i carry MRSA I am i at high risk for developing IS if i get seasonal influenza

Evaluating Language Models

Did adaptation make the LM better?

Best way: plug into ASR system, see how affects WER compared to base LM
Is there something cheaper that predicts WER well?

- Perplexity (PP) of heldout or test data (needs only text).
- Doesn't predict performance well *across* LM types.
- But does within single LM type!
- Has theoretical significance.

Performance of Domain Adapted LMs

Depending on the domain and the amount of in-domain text available, performance improvements in the range of 15-50% relative can be seen.

Domain	**Base LM % WER**	**Adapted LM % WER**
Health care	20.1	9.8
Call center	26.6	22.7

Results presented are on in-house, customer data

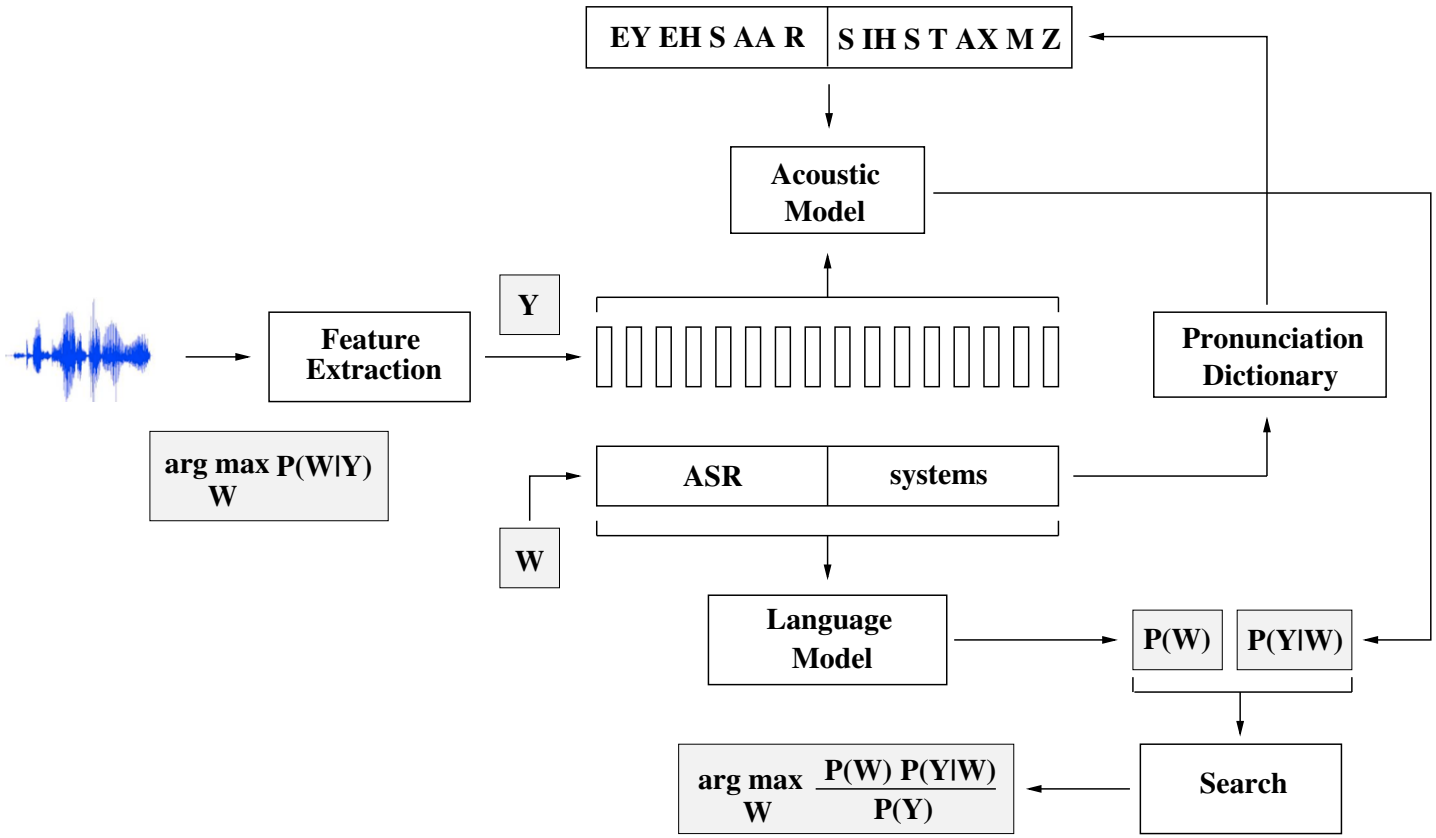
Suggested Combined Recipe for New Domains

1. Obtain relevant in-domain text and include new words to the ASR system's lexicon
2. Using semantically similar or other relevant terms, derive additional text relevant to the domain
3. Build a domain adapted LM using all the above resource
4. Derive transcripts for the in-domain audio (supervised or unsupervised)
5. Use the domain adapted LM to obtain HMM state/character level alignments of the in-domain audio (unsupervised) or the closed LM from reference transcripts for the same
6. Adapt the acoustic model using the audio, transcripts and the domain-adapted LM
7. You now have domain-dependent models that can be used in an ASR system

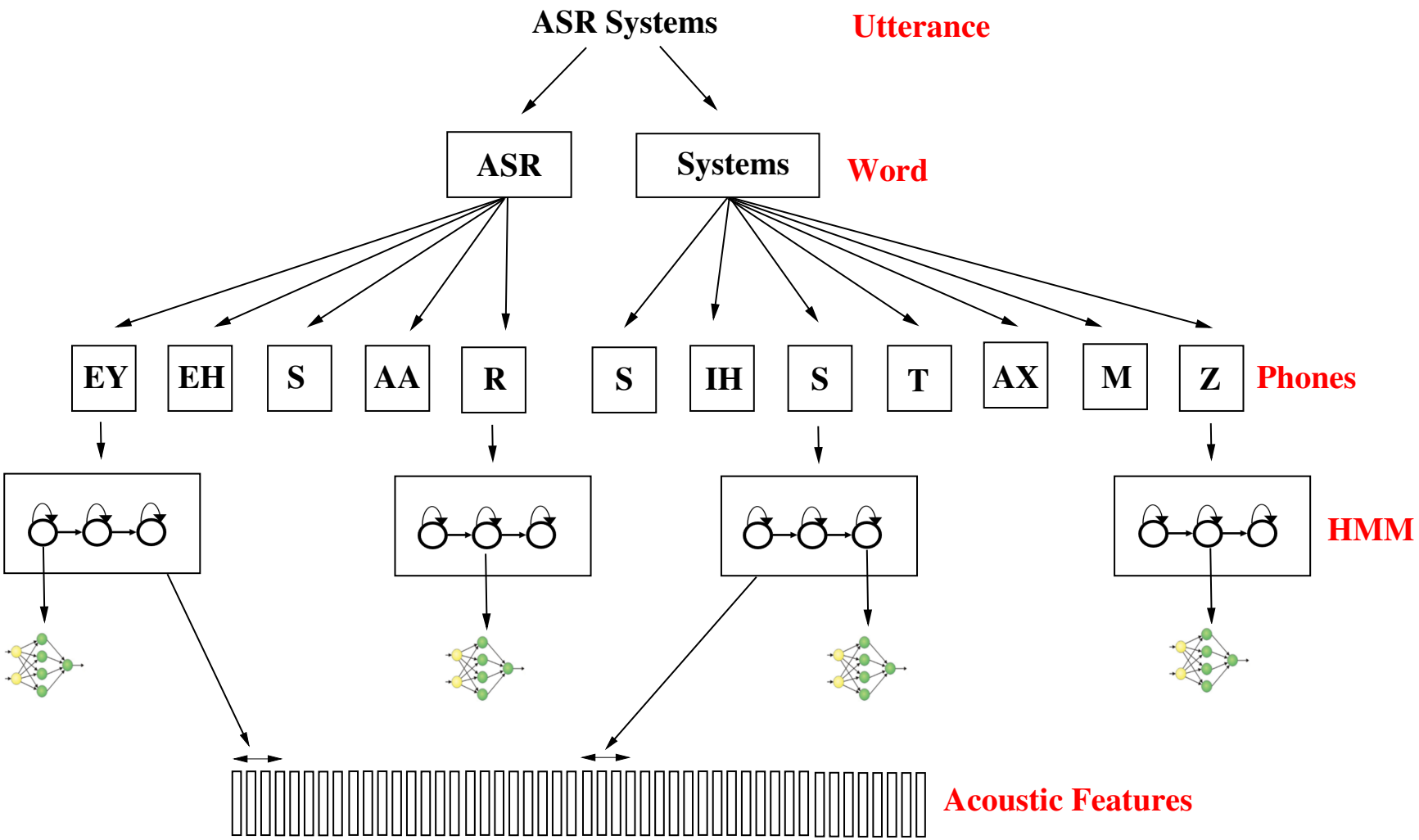
Research Directions and New Modeling Techniques

Challenges

Are we there yet?



Is that it?


ASR Systems
Utterance
ASR
Systems
Word
EY
EH
S
AA
R
S
IH
S
T
AX
M
Z
Phones
HMM
Acoustic Features

Challenges

- What could be wrong?
- We are using a solid statistical framework
- We know how to extract "features"
- We know how to learn models
- We know how to search for the best path
- So we must be doing the right thing!

Really? Let's think about this some more!

- Given: an observation (ADC, FFT, ...)
  $\boldsymbol{X} = \boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_T$
- Wanted: the corresponding word sequence
  $W = w_1, w_2, ..., w_m$
- Search for the most likely word sequence W'
- Fundamental Equation of ASR:
  $W' = \arg\max_W P(W|\boldsymbol{X}) = \arg\max p(\boldsymbol{X}|W)P(W)$ (Bayes)
- $p(\boldsymbol{X}|W)$ is the "acoustic model"
- $P(W)$ is the "language model"
- And how are we evaluating this?

Word Error Rate

- #errors / #words (in reference)
- (#insertions + #substitutions + #deletions) / #words
- Example:

Reference		SHOW	ME	THE	INTERFACE
Hypothesis	I	SHOW	ME		FACE
Alignment	I			D	S

- WER=3/4 = 75%
- So? Did anyone see this in the fundamental equation? $P(W|\boldsymbol{X})$, anyone?

Fundamentals revisited

- If we are optimizing for $P(W|\boldsymbol{X})$, we are optimizing for the *sentence* error rate, $\langle P(W|\boldsymbol{X})\rangle$
- But we want $P(\langle w_i\rangle|\boldsymbol{X})$ for $W = w_1, w_2, ..., w_m$ (for one sentence)
- Clearly, this is not the same thing!
- However, in practice, it is not complete uncorrelated either
  - $\rightarrow$ And there are ways to deal with this (confusion networks, discriminative training)
- Similarly, for spoken term detection, dialog systems, etc.

The speech recognition conundrum

- We want to optimize word error rate (or whatever)

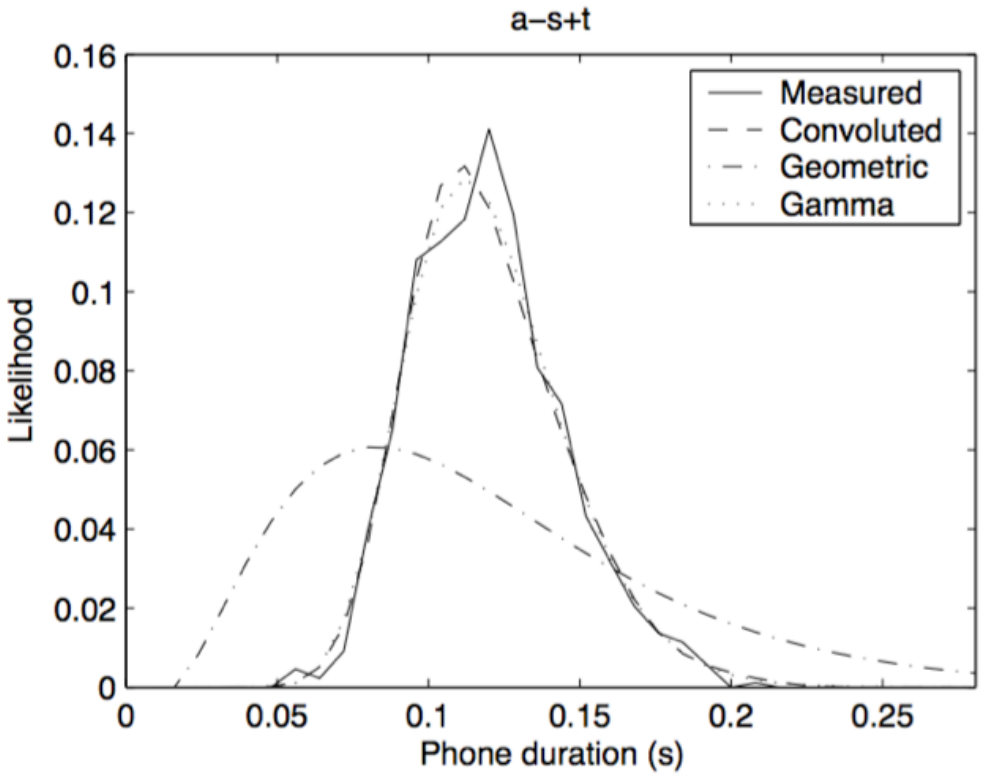
- We optimize the acoustic model for (frame-)likelihood (or cross-entropy, or MPE, sMBR, ...)
- We train a language model for perplexity
- And then we search for the most likely path (a "sentence")
- And this works?

The speech recognition conundrum - It's ugly!

Do we really need Hidden Markov Models?

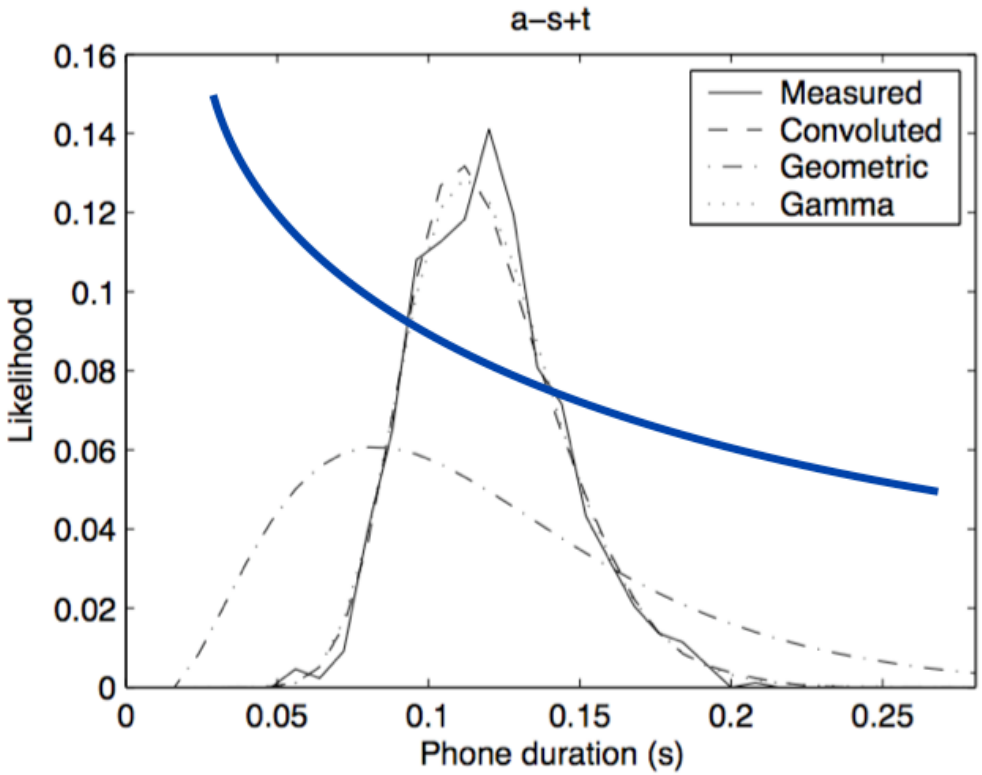
- We know all the theory of why HMMs are good for modelling time series
- We have #states, initial probability distribution, emission probabilities, transition probabilities, and the alphabet to define
- Many different "types" of parameters that we can play with, until we believe we have a good fit
- In practice, we already neglect transition probabilities
- The probability of staying in a state decays exponentially over time

Observed Phone Durations



From [PK04]

Phone Durations with a single-state HMM



- A single HMM state has an exponentially decreasing duration distribution.
- A typical tri-state topology (-B -M -E states) at least has a "maximum" peak for the expected duration of the whole phone

Duration Modelling with HMMs

- A tri-state topology (-b -m -e states) at least has a "maximum" peak for the expected duration of the whole phone
- It is still quite arbitrary - we have plosives, short vowels, long vowels, ...
- **In short**: *let's not make explicit model assumptions that aren't really justified, and that we cannot train the parameters for*
  - (I hope I am not insulting anyone, but:) duration modeling has never shown worthwhile, consistent gains
- **Maybe**, *we should not be using HMMs after all*

Ok, so what do we do?

- There are a few more problems (e.g., hyper-articulation)
- Speech recognition "works"
- Airplanes don't fly the same way birds fly
- We may be fine
- Still, what could we do?

Let's take a step back

- What are we really trying to achieve?
- We want to translate a sequence of observations to a sequence of words
- And ideally, we want to use a single optimality criterion for that
- In 2016? In San Francisco?

There must be a Deep Learning solution for this!

Sequence to Sequence Learning with Neural Networks

Ilya Sutskever
Google
ilyasu@google.com

Oriol Vinyals
Google
vinyals@google.com

Quoc V. Le
Google
qvl@google.com

Abstract

Deep Neural Networks (DNNs) are powerful models that have achieved excellent performance on difficult learning tasks. Although DNNs work well whenever large labeled training sets are available, they cannot be used to map sequences to sequences. In this paper, we present a general end-to-end approach to sequence learning that makes minimal assumptions on the sequence structure. Our method uses a multilayered Long Short-Term Memory (LSTM) to map the input sequence to a vector of a fixed dimensionality, and then another deep LSTM to decode the target sequence from the vector. Our main result is that on an English to French translation task from the WMT-14 dataset, the translations produced by the LSTM achieve a BLEU score of 34.8 on the entire test set, where the LSTM's BLEU score was penalized on out-of-vocabulary words. Additionally, the LSTM did not have difficulty on long sentences. For comparison, a phrase-based SMT system achieves a BLEU score of 33.3 on the same dataset. When we used the LSTM to rerank the 1000 hypotheses produced by the aforementioned SMT system, its BLEU score increases to 36.5, which is close to the previous state of the art. The LSTM also learned sensible phrase and sentence representations that are sensitive to word order and are relatively invariant to the active and the passive voice. Finally, we found that reversing the order of the words in all source sentences (but not target sentences) improved the LSTM's performance markedly, because doing so introduced many short term dependencies between the source and the target sentence which made the optimization problem easier.

Easy, isn't it?

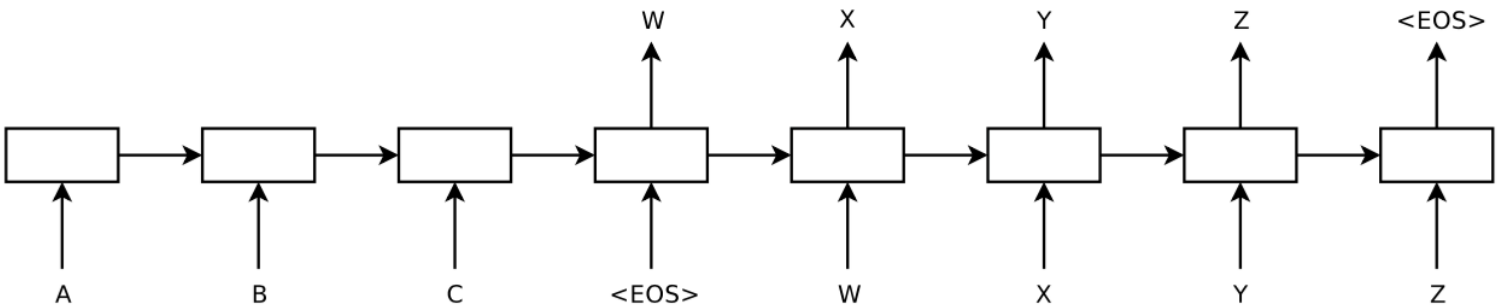


Figure 1: Our model reads an input sentence "ABC" and produces "WXYZ" as the output sentence. The model stops making predictions after outputting the end-of-sentence token. Note that the LSTM reads the input sentence in reverse, because doing so introduces many short term dependencies in the data that make the optimization problem much easier.

$$\rightarrow \text{SPEECH} \mapsto \text{TXET} \rightarrow$$

End-to-end models with attention

- "Sequences" in speech are much longer than in text (even at 30Hz)
- The "encoder"'s task is therefore much harder
- However there is no re-ordering going on (unlike in Machine Translation)
- So, "attention" mechanisms are well suited to recover from decoding errors
  - Imagine the attention mechanism "guiding" an RNN language model

End-to-end models with attention

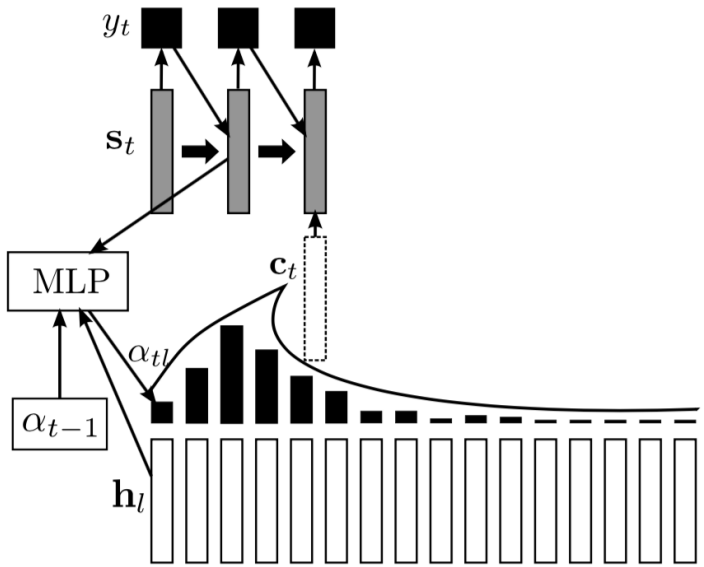


Fig. 3. Schematic representation of the Attention-based Recurrent Sequence Generator. At each time step t, an MLP combines the hidden state $\mathbf{s}_{t-1}$ with all the input vectors $\mathbf{h}_l$ to compute the attention weights α_{tl}. Subsequently, the new hidden state $\mathbf{s}_t$ and prediction for output label y_t can be computed.

From [BCS+15]

And this works? What's the WER?

- According to "Listen, Attend and Spell" ([CJLV15]):
  - Google says they're close on their task (10.3% vs. 8.0%)
  - 1000s of hours of training data
  - "Insane" amount of computation [personal communication]
- Other work emerging, still showing higher word error rates
- But this is probably just the tip of the ice-berg.
  - If you can go from images to text, you can surely go from speech to text
  - Optimization for other criteria and joint learning certainly possible
- What have we gained?
  - There are almost no explicit assumptions in our model
  - Everything becomes a Deep Learning hyper-parameter

So, why did we not start here?

- Problem with end-to-end learning? It is end-to-end!
- How to go from language to another, from one domain to the next?
- Who knows. Topic for a NIPS 2020 Tutorial (or maybe 2018)
  - Condition the decoder on something?
  - Find clever embeddings?
  - ...?

Does it end here?

- No!!!
- Alternatives? Still (somewhat) end-to-end, neural?
- Connectionist Temporal Classification & friends

Connectionist Temporal Classification

- We want to stay sequence-to-sequence
  - Do not bring back any of the "traditional" ballast
  - In particular the CD-HMM complexity
- Separate the acoustics and the language model
  - That's just convenient and pragmatic (even though it's maybe no longer strictly end-to-end)
  - So we want an acoustic model that assumes neighboring units to be independent
- We don't need re-ordering and stuff
  - We don't even really care about time information
  - Speech is continuous and dynamic, maybe there should not be any lasting "states"
- Remember what we said about flat start?
  - Maybe we can put all this in the model?
  - And not deal with frame likelihoods etc. in too much detail

Connectionist Temporal Classification (CTC)

- Alex Graves (2006) described the "CTC" loss function
  - Defined over a label sequence (length M)
  - Sum over all possible frame alignments (length T) permitted for label sequence using Forward-Backward
- Plays well with RNN or LSTM neural network models
- CTC introduces a new symbol: blank (-)
  - "No output"
  - Do not confuse with "silence"
- Most of the time, the network will output (-)
  - "Sparse" model, not dense in time
  - Class im-balance not a problem in a connectionist architecture
  - As long as the target symbols appear from time to time

Connectionist Temporal Classification

- How does this work?
- The probability of recognizing a "label" is the sum of all permitted paths (over "frames") that correspond to that "label"
- Example: label (sequence) ***l*** is A
  - Permitted paths π for A are A, -A, --AAA, ...
  - But not A-A
- Example: label sequence ***l*** is AB
  - Permitted paths π for AB are e.g. AB, ----AAAB--, -A-BB---, ...
  - AA has to be A-A, but AB can be AB or A-B
- A grammar that collapses repetitions and deletes blanks
  - Non bijective map $\mathcal{B}$ between paths π and label sequences ***l***

CTC Loss Function

$$p(\boldsymbol{l}|\boldsymbol{X}) = \sum_{\pi \in \mathcal{B}^{-1}(\boldsymbol{l})} p(\pi|\boldsymbol{X}) \tag{1}$$

$$p(\pi|\boldsymbol{X}) = \prod_{t=1}^{T} y_{\pi_t}^t, \forall \pi \in L'^T \tag{2}$$

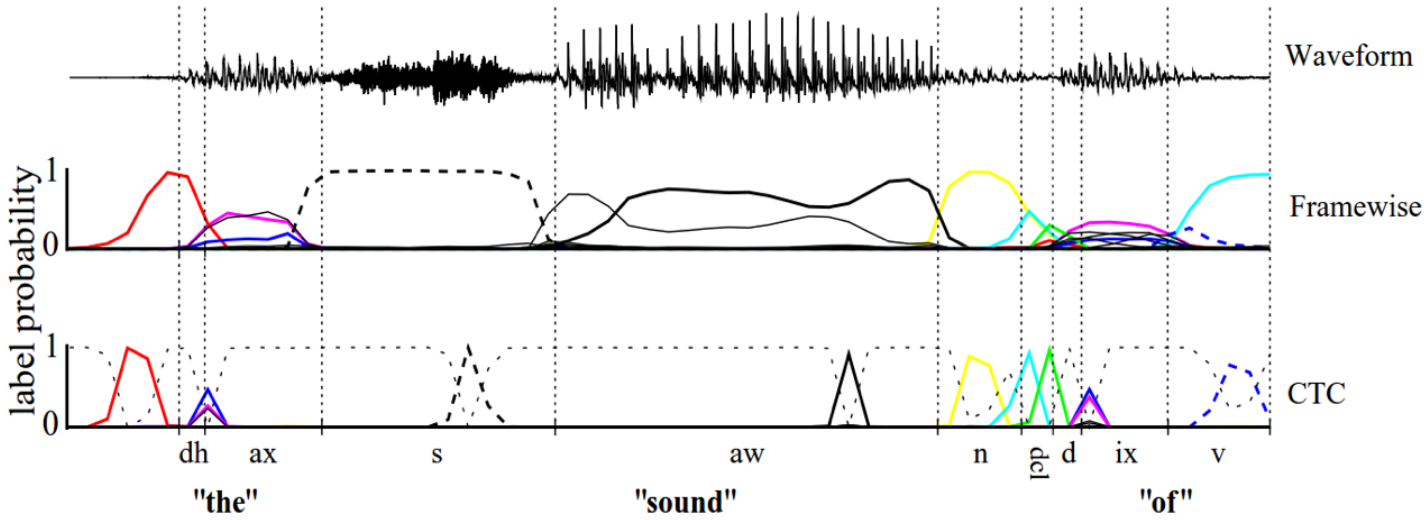
- $y_{\pi_t}^t$ is the output of the network at time t (according to the path)
- The likelihood of a labeling $p(\boldsymbol{l}|\boldsymbol{X})$ can be computed efficiently using Fwd-Bwd over all permitted paths, using prefixes/ postfixes
- Also introduce an augmented label sequence $\boldsymbol{l}'$ which includes optional blanks

CTC Loss Function

$$O^{ML}(S, \mathcal{N}_W) = -\sum_{(x,z)\in S} ln(p(z|x)) \tag{3}$$

- Loss function is neg-log prob of correctly labeling all observation/ labeling pairs $(x, z) \in S$
- $y = \mathcal{N}_w(x)$ is the mapping that the network induces
- Consider each sequence pair independently, indexed by t
- Interestingly, the error can be backpropagated through this easily

What does this look like?



From [GFGS06]

Efficient Decoding with CTC

- Originally, it was not clear how these things could be decoded efficiently
- Turns out, a wFST (weighted Finite State Transducer) framework works just fine [MGM15, SSRB15]
- Need to carefully design the T transducers for the tokens, which however replace the H (HMM) and C (context) transducers in conventional HMM-based systems
- Get class posteriors directly from label counts
- Advantage(s) at comparable accuracy
  - CTC system is much smaller and usually faster during decoding
  - Can use 30ms frame rate and works well with graphemic units
  - N.B.: some of these findings might not be due to CTC, but to LSTMs, and may be back-ported

Research Topics

- What does this mean for spoken term detection?
  - The lack of reliable timings is a problem
- Do end-to-end approaches work in low resource scenarios?
  - 10s of hours of training data, rather than 1000s?
- Can we make ASR simpler?
- Can we build multi-lingual systems, use multi-task training?
- Can we use this to improve phone recognition

New Ideas

- Combine CTC and attention-based models
  - Use CTC as a front-end to attention-based models
  - A la language independent feature extractors?
  - Makes it easier for an encoder/ decoder model to learn
  - Across languages?
- Integrate non-auditory information
  - Multimedia data is very rich, e.g. video
  - Extract objects, scenes, speaker info from the visual part
  - Use it to bootstrap, adapt, condition, or otherwise improve acoustic and/ or language models
- Go beyond the speech-to-text paradigm
  - Maybe a verbatim transcription is not always what we want
  - Combine different information sources intelligently to give us audio-to-meaning in a new domain or language

Hands-On Experience with Virtual Machines

Practicalities

- We want to give you hands-on experience with building ASR systems
- You will be able to train a system on a Babel language (most likely 201 Haitian)
- You can then experiment with other Babel languages, or port the system to other domains
- To facilitate experimentation, we will distribute a Virtual Machine (VM)
- Read on to see how you can prepare

Virtual Machines and Tools

- Think of a VM as a "virtual" computer, in our case running Linux
- VMs allow sharing reproducible experiments easily
- https://github.com/srvk, http://speechkitchen.org as repositories
- https://www.vagrantup.com/ to build VMs
- https://www.virtualbox.org/ to run VMs (along with https://aws.amazon.com/)
- An "image" is a computer when it is turned off, it becomes an "instance" when you turn it on

Exercises

- We will share a Vagrantfile, plus an image on AWS (most likely), and/ or a Virtualbox OVA (less likely)
- Your best bet is to run the exercise on AWS
- So, you may want to sign up for an account first (https://aws.amazon.com/getting-started/)
- Familiarize yourself with how to start a Linux VM on "EC2" using a pre-configured Amazon Machine Image (AMI)
- Training a DNN-based recognizer on a GPU will cost some money, but the cost should not be dramatic
- Once you reproduced the basics, you can continue on AWS, or you can migrate to your own infrastructure

Eesen

- We will use the "Eesen" toolkit (https://github.com/srvk/eesen) for end-to-end speech recognition
- It is based on Kaldi (http://kaldi-asr.org/), but a bit smaller and easier to handle
- More details to follow

Conclusions

- It works.
- Sort of.
- Extra Bavariam nulla vita, et si est vita, non est ita.

Thank You!

Any Questions?

Acknowledgment and disclaimer

This work is supported in part by the Intelligence Advanced Research Projects Activity (IARPA) via Department of Defense U.S. Army Research Laboratory (DoD/ARL) contract number W911NF-12-C-0012. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon.

Disclaimer: The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of IARPA, DoD/ARL, or the U.S. Government.

Babel resources

This work uses the following Babel data packs.

IARPA-babel102b-v0.5a
IARPA-babel201b-v0.2b
IARPA-babel203b-v3.1a
IARPA-babel206b-v0.1e
IARPA-babel207b-v1.0c
IARPA-babel303b-v1.0a
IARPA-babel305b-v1.0b
IARPA-babel306b-v2.0c
IARPA-babel307b-v1.0b
IARPA-babel401b-v2.0b
IARPA-babel402b-v1.0b
IARPA-babel403b-v1.0b
IARPA-babel404b-v1.0a

References

[BCS+15] D. Bahdanau, J. Chorowski, D. Serdyuk, P. Brakel, and Y. Bengio, "End-to-end attention-based large vocabulary speech recognition," *CoRR*, vol. abs/1508.04395, 2015. [Online]. Available: http://arxiv.org/abs/1508.04395

[CJLV15] W. Chan, N. Jaitly, Q. V. Le, and O. Vinyals, "Listen, attend and spell," *arXiv preprint arXiv:1508.01211*, 2015.

[GFGS06] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber, "Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks," in *Proceedings of the 23rd international conference on Machine learning*. ACM, 2006, pp. 369–376.

[GKB09] F. Grézl, M. Karafiát, and L. Burget, "Investigation into bottle-neck features for meeting speech recognition," in *Proc. Interspeech*, 2009.

[KSS12] B. Kingsbury, T. N. Sainath, and H. Soltau, "Scalable minimum Bayes risk training of deep neural network acoustic models using distributed Hessian-free optimization," in *Proc. Interspeech*, 2012.

[LHE08] K. Laskowski, M. Heldner, and J. Edlund, "The fundamental frequency variation spectrum," in *Proc. FONETIK*. Dept. of Linguistics, University of Gothenburg, 2008.

[MGM15] Y. Miao, M. Gowayyed, and F. Metze, "EESEN: End-to-End Speech Recognition using Deep RNN Models and WFST-based Decoding," in *Proc. Automatic Speech Recognition and Understanding Workshop (ASRU)*. Scottsdale, AZ; U.S.A.: IEEE, Dec. 2015, https://github.com/srvk/eesen.

[PK04] J. Pylkkönen and M. Kurimo, "Duration modeling techniques for continuous speech recognition," in *In Proceedings of the 8th International Conference on Spoken Language Processing*, 2004.

[SKS+13] T. N. Sainath, B. Kingsbury, V. Sindhwani, E. Arisoy, and B. Ramabhadran, "Low-rank matrix factorization for deep neural network training with high-dimensional output targets," in *Proc. ICASSP*, 2013.

[SLCY11] F. Seide, G. Li, X. Chen, and D. Yu, "Feature engineering in context-dependent deep neural networks for conversational speech transcription," in *Proc. ASRU*, 2011.

[SLF+08] S. Scanzio, P. Laface, L. Fissore, R. Gemello, and F. Mana, "On the use of a multilingual neural network front-end," in *Proc. Interspeech*, 2008.

[SSRB15] H. Sak, A. Senior, K. Rao, and F. Beaufays, "Fast and accurate recurrent neural network acoustic models for speech recognition," *arXiv preprint arXiv:1507.06947*, 2015.

[VSGK13] S. Virpioja, P. Smit, S.-A. Grönroos, and M. Kurimo, "Morfessor 2.0: Python implementation and extensions for Morfessor baseline," Aalto University, Helsinki, Finland, Tech. Rep. SCIENCE + TECHNOLOGY, 25/2013, 2013, iSBN 978-952-60-5501-5.

[WM14] Y. Wang and F. Metze, "An in-depth comparison of keyword specific thresholding and sum-to-one score normalization," in *Proc. Interspeech*, 2014, https://www.cs.cmu.edu/~fmetze/interACT/Publications_files/publications/KST_vs_STO.pdf.