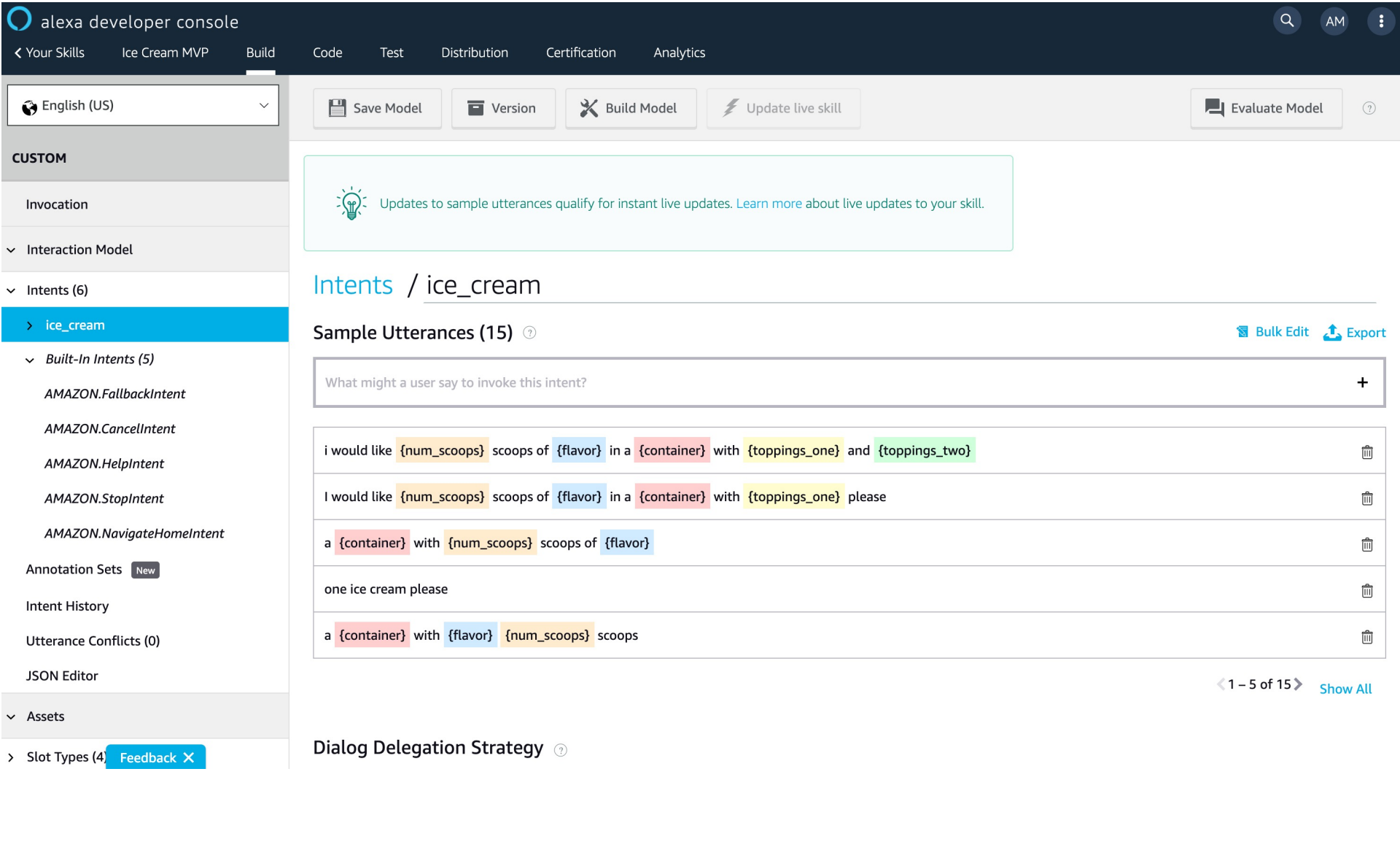
CS 224S / LINGUIST 285
Spoken Language Processing

Andrew Maas
Stanford University
Spring 2022

Lecture 7: Course Project and History

Homework 2 due Monday 11:59pm



Outline for Today

- Some history of recognition and synthesis
- Course project overview
- Q&A on course project and/or specific project ideas

Arc of recent history

- In 2014:
  - ASR, TTS, dialog all used specialized, hard to build modeling approaches
  - Industry application of SLU systems limited
- Today:
  - ASR, TTS, dialog all use deep learning approaches. Less specialized and better performance
  - Spoken language systems are everywhere!
  - New tools enable building full systems

History: foundational insights 1900s-1950s

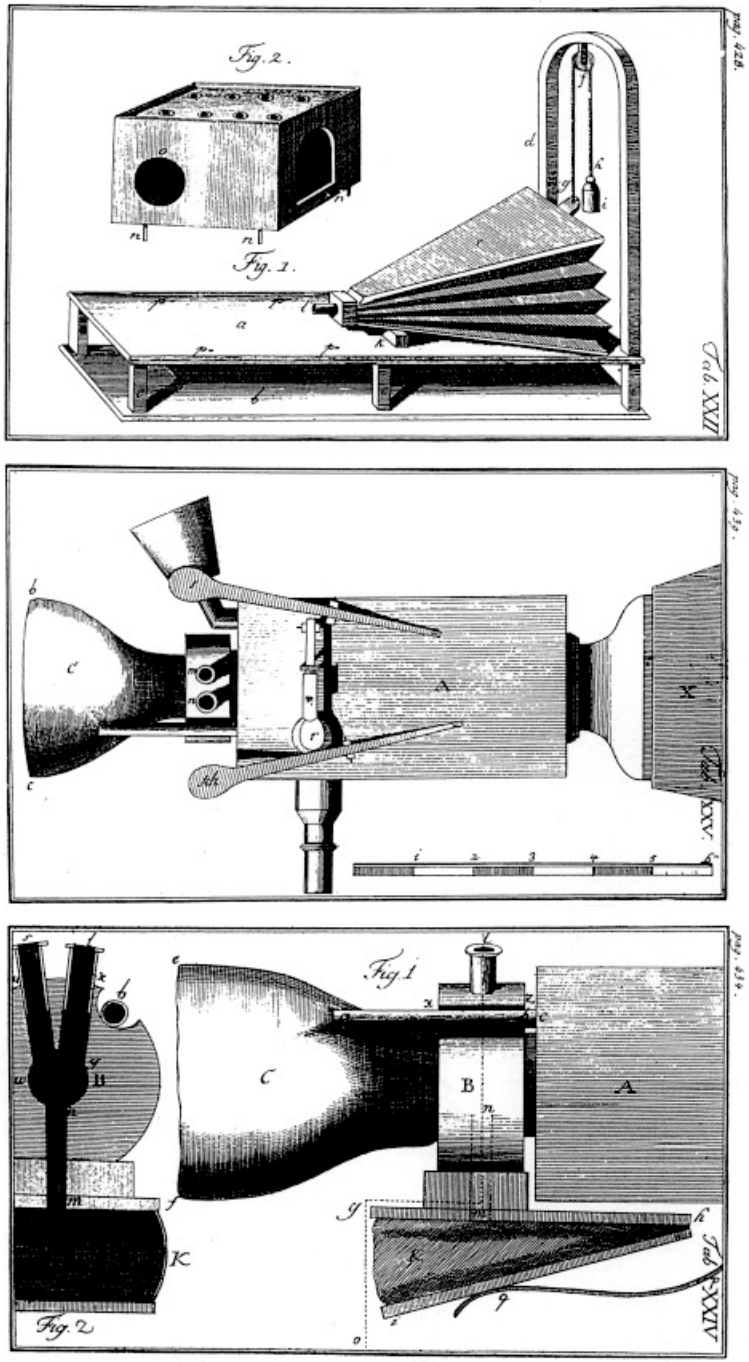
- Automaton:
  - Markov 1911
  - Turing 1936
  - McCulloch-Pitts neuron (1943)
    - http://marr.bsee.swin.edu.au/~dtl/het704/lecture10/ann/node1.html
    - http://diwww.epfl.ch/mantra/tutorial/english/mcpits/html/
  - Shannon (1948) link between automata and Markov models
- Human speech processing
  - Fletcher at Bell Labs (1920's)
- Probabilistic/Information-theoretic models
  - Shannon (1948)

Speech synthesis is old!

- Pictures and some text from Hartmut Traunmüller's web site:
  - http://www.ling.su.se/staff/hartmut/kemplne.htm
- Von Kempeln 1780 b. Bratislava 1734 d. Vienna 1804
- Leather resonator manipulated by the operator to try and copy vocal tract configuration during sonorants (vowels, glides, nasals)
- Bellows provided air stream, counterweight provided inhalation
- Vibrating reed produced periodic pressure wave

Von Kempelen:

- Small whistles controlled consonants
- Rubber mouth and nose; nose had to be covered with two fingers for non-nasals
- Unvoiced sounds: mouth covered, auxiliary bellows driven by string provides puff of air



From Traunmüller's web site

History: Early Recognition

- 1920's Radio Rex
  - Celluloid dog with iron base held within house by electromagnet against force of spring
  - Current to magnet flowed through bridge which was sensitive to energy at 500 Hz
  - 500 Hz energy caused bridge to vibrate, interrupting current, making dog spring forward
  - The sound "e" (ARPAbet [eh]) in Rex has 500 Hz component

History: early ASR systems

- 1950's: Early Speech recognizers
  - 1952: Bell Labs single-speaker digit recognizer
    - Measured energy from two bands (formants)
    - Built with analog electrical components
    - 2% error rate for single speaker, isolated digits
  - 1958: Dudley built classifier that used continuous spectrum rather than just formants
  - 1959: Denes ASR combining grammar and acoustic probability

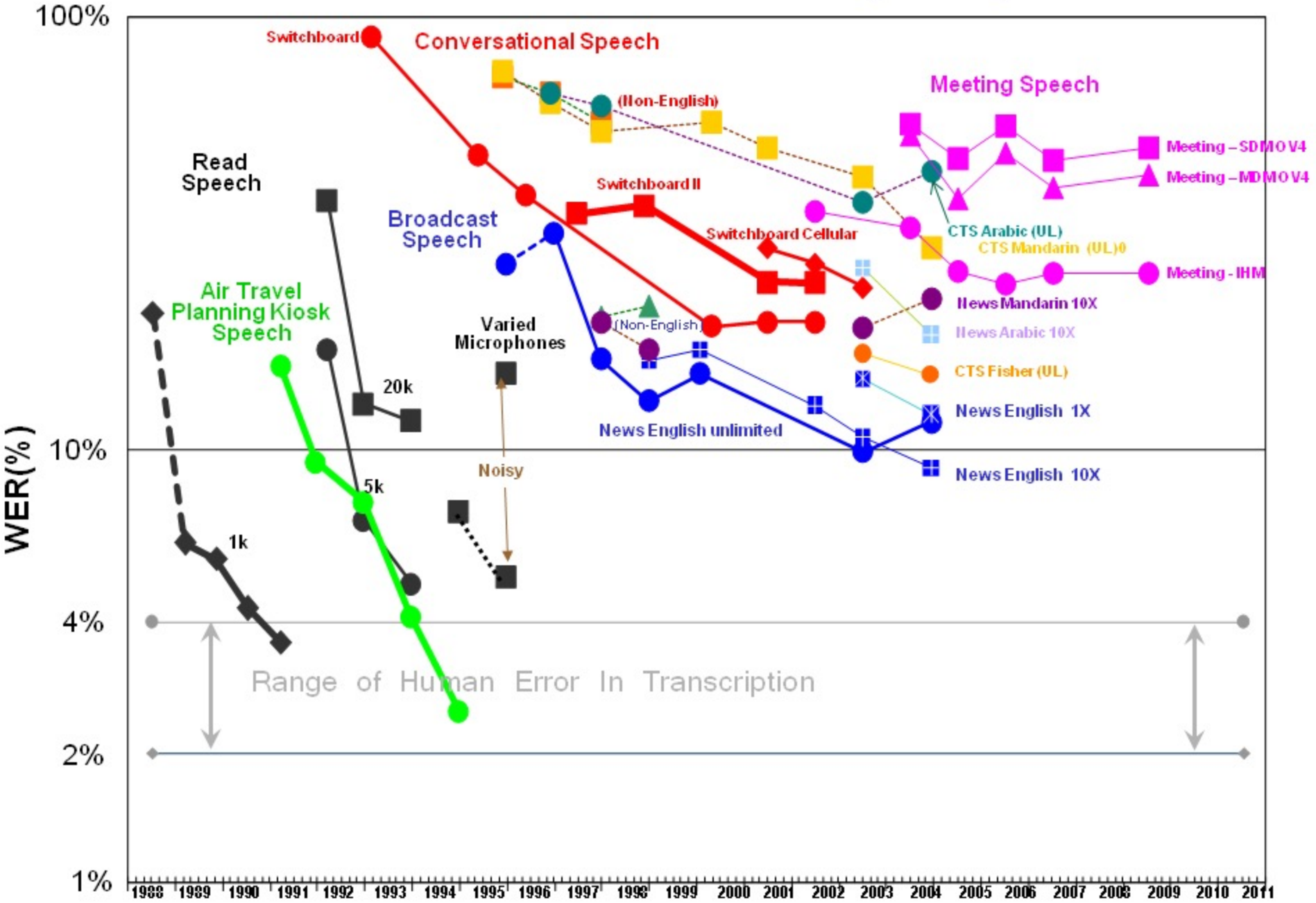
History: early ASR systems

- 1960's
  - FFT - Fast Fourier transform (Cooley and Tukey 1965)
  - LPC - linear prediction (1968)
  - 1969 John Pierce letter "Whither Speech Recognition?"
    - Random tuning of parameters,
    - Lack of scientific rigor, no evaluation metrics
    - Need to rely on higher level knowledge

ASR: 1970's and 1980's

- Hidden Markov Model 1972
  - Independent application of Baker (CMU) and Jelinek/Bahl/Mercer lab (IBM) following work of Baum and colleagues at IDA
- ARPA project 1971-1976
  - 5-year speech understanding project: 1000 word vocab, continous speech, multi-speaker
  - SDC, CMU, BBN
  - Only 1 CMU system achieved goal
- 1980's +
  - Annual ARPA "Bakeoffs"
  - Large corpus collection
    - TIMIT
    - Resource Management
    - Wall Street Journal

NIST STT Benchmark Test History – May. '09



Recent ASR improvements

Hub5'00 Evaluation (Switchboard / CallHome)

(Possibly trained on more data than SWB, but test set = full Hub5

WER (SWB)	WER (CH)	Paper	Published
4.9%	9.5%	An investigation of phone-based subword units for end-to-end speech recognition	April 2020
5.0%	9.1%	The CAPIO 2017 Conversational Speech Recognition System	December 2017
5.1%	9.9%	Language Modeling with Highway LSTM	September 2017
5.1%		The Microsoft 2017 Conversational Speech Recognition System	August 2017

WSJ

(Possibly trained on more data than WSJ.)

WER eval'92	WER eval'93	Paper	Published
5.03%	8.08%	*Humans* Deep Speech 2: End-to-End Speech Recognition in English and Mandarin	December 2015
2.9%		End-to-end Speech Recognition Using Lattice-Free MMI	September 2018
3.10%		Deep Speech 2: End-to-End Speech Recognition in English and Mandarin	December 2015

LibriSpeech

(Possibly trained on more data than LibriSpeech.)

WER test-clean	WER test-other	Paper	Published
5.83%	12.69%	*Humans* Deep Speech 2: End-to-End Speech Recognition in English and Mandarin	December 2015
1.9%	3.9%	Conformer: Convolution-augmented Transformer for Speech Recognition	May 2020
1.9%	4.1%	ContextNet: Improving Convolutional Neural Networks for Automatic Speech Recognition with Global Context	May 2020

https://github.com/syhw/wer_are_we

Outline for Today

- Some history of recognition and synthesis
- **Course project overview**
- Q&A on course project and/or specific project ideas

Course project goals

- A substantial piece of work related to topics specific to this course
- A successful project result:
  - Most of a conference paper submission if academically oriented -or-
  - Functioning system demo. A portfolio item for job interviews, or a system you put into production!
- Reflects deeper understanding of SLP technology than simply applying existing API's for ASR, voice commands, etc.
- *Build something you're proud of*

A successful project

- Course-relevant topic. Proposed experiments or system address a challenging, unsolved SLP problem
- Proposes and executes a sensible approach informed by previous related work
- Performs error analysis to understand what aspects of the system are good/bad
- Adapts system or introduces new hypotheses/components based on initial error analysis
- Goes beyond simply combining existing components / tools to solve a standard problem

Complexity and focus

- SLP systems are some of the most complex in AI
- Example: A simple voice command system contains:
  - Speech recognizer (Language model, pronunciation lexicon, acoustic model, decoder, lots of training options)
  - Intent/command slot filling (some combination of lexicon, rules, and ML to handle variation)
- Get a complete baseline system working by milestone
- Focus on a subset of all areas to make a bigger contribution there. APIs/tools are a great choice for areas not directly relevant to your focus

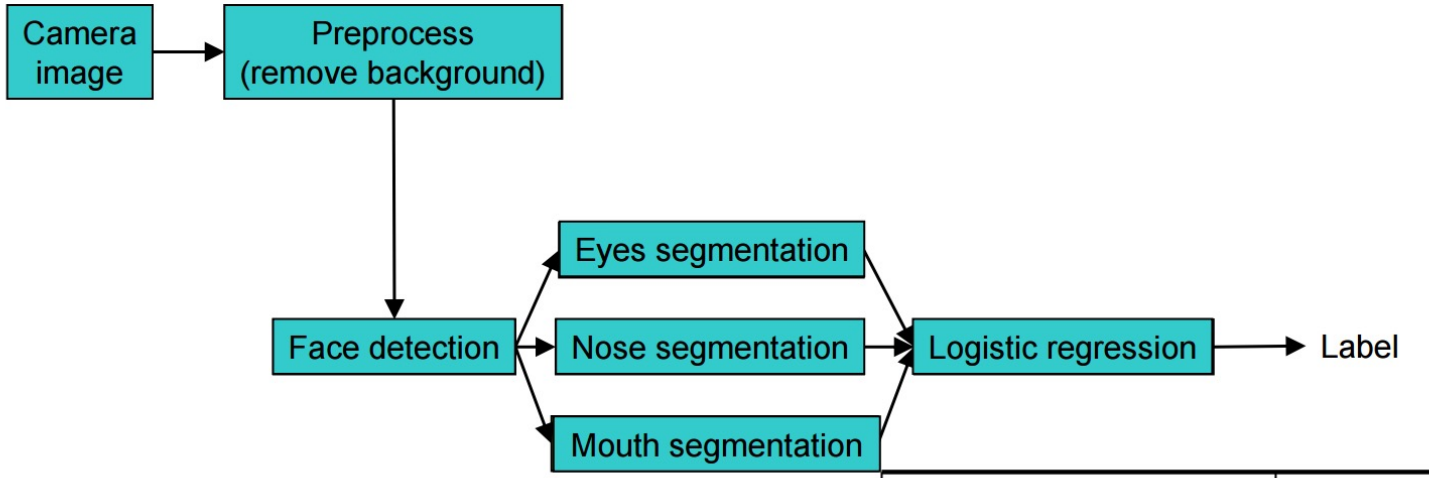
Balancing scale and depth

- Working on “real” scale datasets/problems is a plus…
- But don’t let scale distract from getting to the meat of your technical contribution
- Example:
  - Comparing some neural architectures for end-to-end speech recognition
  - Case 1: Use WSJ. Medium sized corpus, read speech. SOTA error rates ~3%
  - Case 2: Use Switchboard: Large, conversational corpus. SOTA error rates ~15%
- Case 2 stronger overall *if you run the same experiments / error analysis. Don’t let scale prevent “thoughtful loops”*

Thoughtful loops

- A single loop:
  - Try something reasonable with a hypothesis
  - Perform error analysis to investigate your hypothesis
  - Propose a modification / new experiment based on what you find
  - Try it!
  - Repeat above
- A successful project does this at least once
- Scale introduces risk of overly slow loops
- Ablative analysis or oracle experiments are a great way to guide what system component to work on

Oracle experiments



How much error is attributable to each of the components?

Plug in ground-truth for each component, and see how accuracy changes.

Conclusion: Most room for improvement in face detection and eyes segmentation.

Component	Accuracy
Overall system	85%
Preprocess (remove background)	85.1%
Face detection	91%
Eyes segmentation	95%
Nose segmentation	96%
Mouth segmentation	97%
Logistic regression	100%

Slide from Andrew Ng's CS229 lecture on applying ML
http://cs229.stanford.edu/materials/ML-advice.pdf

Ablation experiments

Error analysis tries to explain the difference between current performance and perfect performance.

Ablative analysis tries to explain the difference between some baseline (much poorer) performance and current performance.

E.g., Suppose that you've build a good anti-spam classifier by adding lots of clever features to logistic regression:

- Spelling correction.
- Sender host features.
- Email header features.
- Email text parser features.
- Javascript parser.
- Features from embedded images.

Question: How much did each of these components really help?

Slide from Andrew Ng's CS229 lecture on applying ML
http://cs229.stanford.edu/materials/ML-advice.pdf

Ablation experiments

Simple logistic regression without any clever features get 94% performance.

Just what accounts for your improvement from 94 to 99.9%?

Ablative analysis: Remove components from your system one at a time, to see how it breaks.

Component	Accuracy	
Overall system	99.9%	
Spelling correction	99.0	
Sender host features	98.9%	
Email header features	98.9%	
Email text parser features	95%	
Javascript parser	94.5%	
Features from images	94.0%	[baseline]

Conclusion: The email text parser features account for most of the improvement.

Slide from Andrew Ng's CS229 lecture on applying ML
http://cs229.stanford.edu/materials/ML-advice.pdf

Pitfalls in project planning

- Data!
  - What dataset will you use for your task?
  - If you need to collect data, why? Understand that a project with a lot of required data collection creates high risk of not being able to execute enough loops
  - Do you really need to collect data? Really?
- Overly complex baseline system
- Relying on external tools to the point that connecting them becomes the entire effort and makes innovation hard
- Off-topic. Could this be a CS 229 project instead?

Deliverables

- All projects
  - Proposal: What task, dataset, evaluation metrics and approach outline?
  - Milestone: Have you gotten your data and built a baseline for your task?
  - Final paper: Methods, results, related work, conclusions. Should read like a conference paper
- Audio/Visual material
  - Include links to audio samples for TTS. Screen capture videos for dialog interactions (spoken dialog especially)
  - Much easier to understand your contribution this way than leave us to guess. Even if it doesn't quite work.

Leveraging existing tools

- Free to use any tool, but realize using the Google speech API does not constitute 'building a recognizer'
- Ensure the tool does not prevent trying the algorithmic modifications of interest (e.g. can't do acoustic model research on speech API's)
- Projects that combine existing tools in a straightforward way should be avoided
- Conversely, almost every project can *and should* use some form of tool:
  - PyTorch, Tensorflow, speech API, language model toolkit, Alexa, Kaldi, etc.
- Use tools to focus on your project hypotheses

Error analysis with tools

- Project writeup / presentation should be able to explain:
  - What goal does this tool achieve for our system?
  - Is the tool a source of errors? (e.g. oracle error rate for a speech API)
  - How could this tool be modified / replaced to improve the system? (maybe it is perfect and that's okay)
- As with any component, important to isolate sources of errors
- Work with tools in a way that reflects your deeper understanding of what they do internally (e.g. n-best lists)

Sample of tools and APIs

- Speech APIs: Google, IBM, Microsoft all have options
  - Varying levels of customization and conveying n-best
- Speech synthesis APIs: same as speech + Festival
- Slack or Facebook for text dialog interfaces
  - Slack allows downloading of historical data which could help train systems
  - Howdy.ai / botkit for integration
- Dialog / intent recognition tools
  - Wit.ai, API.ai, Alexa Skills Kit, Uber Plato
- Many new open source deep learning models for ASR/TTS. Some with pre-trained models

Sample project archetypes

- Previous projects: http://web.stanford.edu/class/cs224s/project/

Research trend: Foundation model features

- Large, pre-trained deep learning models provide useful features for many spoken tasks
- Lots to explore in this area in ASR, TTS, dialog, and multi-modal
- Pre-trained models great for rapid progress on projects and smaller training sets
- Popular models with available features:
  - Wav2Vec 2.0
  - HuBERT
  - HuggingFace has several more!

Research trend: Foundation model features

TABLE II: An overview of the recent audio self-supervised learning methods. The "speech" column distinguishes whether a method addresses speech tasks or for general purpose audio representations. The "framework" type refers to Figure 1.

Model	Speech	Input format	Framework	Encoder	Loss	Inspired by
LIM [36]	✔	raw waveform	(d)	SincNet	BCE, MINE or NCE loss	SimCLR
COLA [36]	✗	log mel-filterbanks	(d)	EfficientNet	InfoNCE loss	SimCLR
CLAR [33] (semi)	✗	raw waveform log mel-spectrogram	(d)	1D ResNet-18 ResNet-18	NT-Xent + cross-entropy	SimCLR
Fonseca et al. [36]	✗	log mel-spectrogram	(d)	ResNet, VGG, CRNN	NT-Xent loss	SimCLR
Wang et al. [88]	✗	raw waveform + log mel-filterbanks	(d)	CNN ResNet	NT-Xent loss + cross-entropy	SimCLR
BYOL-A [89]	✗	log mel-filterbanks	(b)	CNN	MSE loss	BYOL
Speech2Vec [48]	✔	mel-spectrogram	(a)	RNN	MSE loss	Word2Vec
Audio2Vec [91]	✔✗	MFCCs	(a)	CNN	MSE loss	Word2Vec
Carr [67]	✔	MFCCs	(a)	Context-free network	Fenchel-Young loss	-
Ryan [68]	✗	constant-Q transform spectrogram	(a)	AlexNet	Triplet loss	- -
Mockingjay [92]	✔	mel-spectrogram	(a)	Transformer	L1 loss	BERT
TERA [93]	✔	log mel-spectrogram	(a)	Transformer	L1 loss	BERT
Audio ALBERT [94]	✔	log mel-spectrogram	(a)	Transformer	L1 loss	BERT
DAPC [95]	✔	spectrogram	(a)	Transformer	Modified MSE loss + orthogonality penalty	BERT
PASE [96]	✔	raw waveform	(a)	SincNet + CNN	L1, BCE loss	BERT
PASE+ [97]	✔	raw waveform	(a)	SincNet + CNN + QRNN	MSE, BCE loss	BERT
CPC [40]	✔	raw waveform	(a)	ResNet + GRU	InfoNCE loss	-
CPC v2 [59]	✔	raw waveform	(a)	ResNet + Masked CNN	InfoNCE loss	-
CPC2 [98]	✔	raw waveform	(a)	ResNet + LSTM	InfoNCE loss	-
Wav2Vec [84]	✔	raw waveform	(a)	1D CNN	Contrastive loss	-
VQ-Wav2Vec [85]	✔	raw waveform	(a)	1D CNN + BERT	Contrastive loss	BERT
Wav2Vec 2.0 [81]	✔	raw waveform	(a)	1D CNN + Transformer	Contrastive loss	BERT
HuBERT [99]	✔	raw waveform	(c)	1D CNN + Transformer	Contrastive loss	BERT

(Liu *et al*, 2022)

Research trend: Convergence of text NLP and spoken NLP

- Can you simplify spoken language modeling to close the text/spoken gap?
- Recipe for NLP tasks:
  - Transformer models
  - Self-supervised pre-training on large data
  - Fine tune / adapt to particular tasks

(Liu *et al*, 2022)

Dialog systems

- Build a dialog system for a task that interests you (bartender, medical guidance, chess)
- *Must be multi-turn*. Not just voice commands or single slot intent recognizers
- Evaluation is difficult, likely will have to collect any training data yourself
- Don’t over-invest in knowledge engineering
- Lots of room to be creative and design interactions to hide system limitations
- More difficult to publish smaller scale systems, but make for great demos / portfolio items

Speech recognition

- Benchmark corpus (WSJ, Switchboard, noisy ASR on CHIME, Librispeech)
- Establish a baseline system (ok to use pre-trained)
- Template very amenable to publication in speech or machine learning conferences
- Can be very difficult to improve on state of the art. Systems can be cumbersome to train
- Lots of algorithmic variations to try
- Successful projects do not need to improve on best existing results
- Adapting pre-trained neural approaches to new domains/tasks is great!

Speech synthesis

- Blizzard challenge provides training data and systems for comparison
- Evaluation is difficult. No single metric
- Matching state of the art can be very tedious signal processing
- Open realm of experiments to try, especially working to be expressive or improve prosody
- Relatively large systems for SOTA. Recent deep learning approaches for easier testing

Extracting information from speech

- Beyond transcription, understanding emotion, accent, or mental state (intoxication, depression, Parkinson's etc.)
- Very dataset dependent. How will you access labeled data to train a system?
- Can't be just a classifier. Need to use insights from this course or combine with speech recognition
  - Exploring foundation model features is acceptable
- Should be spoken rather than just written text
- Often most exciting when the dataset/task is meaningful

Deep learning approaches

- Active area of research for every area of SLP
- Beware:
  - Do you have enough training data compared to the most similar paper to your approach?
  - Do you have enough compute power / GPUs?
  - How long will a single model take to train? Think about your time to complete one ‘loop’
- *Ensure you are doing SLP experiments not just tuning neural nets for a dataset*
- Hot area for academic publications

Summary

- Have fun
- Build something you're proud of
- Post on Ed or use Andrew's Thursday office hours for early project feedback
- Proposals due May 3

Discussion/Questions

Appendix

Dialogue System Evaluation

- Always two kinds of evaluation
  - Extrinsic: embedded in some external task
  - Intrinsic: evaluating the component as such

- What constitutes success or failure for a dialogue system?

Reasons for Dialogue System Evaluation

1. A metric to compare systems
   - can't improve it if we don't know where it fails
   - can't decide between two systems without a goodness metric
2. A metric as an input to reinforcement learning:
   - automatically improve conversational agent performance via learning

Evaluation

1. Slot Error Rate for a Sentence

$$\frac{\text{\# of inserted/deleted/subsituted slots}}{\text{\# of total reference slots for sentence}}$$

2. End-to-end evaluation (Task Success)

Evaluation Metrics

"Make an appointment with Chris at 10:30 in Gates 104"

Slot	Filler
PERSON	Chris
TIME	11:30 a.m.
ROOM	Gates 104

Slot error rate: 1/3

Task success: At end, was the correct meeting added to the calendar?

Interaction satisfaction: User ratings. # turns

Task Success

- % of subtasks completed
- Correctness of each questions/answer/error msg
- Correctness of total solution
  - Error rate in final slots
    - Generalization of Slot Error Rate
- Users' perception of whether task was completed

Efficiency Cost

Polifroni et al. (1992), Danieli and Gerbino (1995)
Hirschman and Pao (1993)

- Total elapsed time in seconds or turns
- Number of queries
- Turn correction ration: number of system or user turns used solely to correct errors, divided by total number of turns

Quality Cost

- # of times ASR system failed to return any sentence
- # of ASR rejection prompts
- # of times user had to barge-in
- # of time-out prompts
- Inappropriateness (verbose, ambiguous) of system's questions, answers, error messages

Concept accuracy:

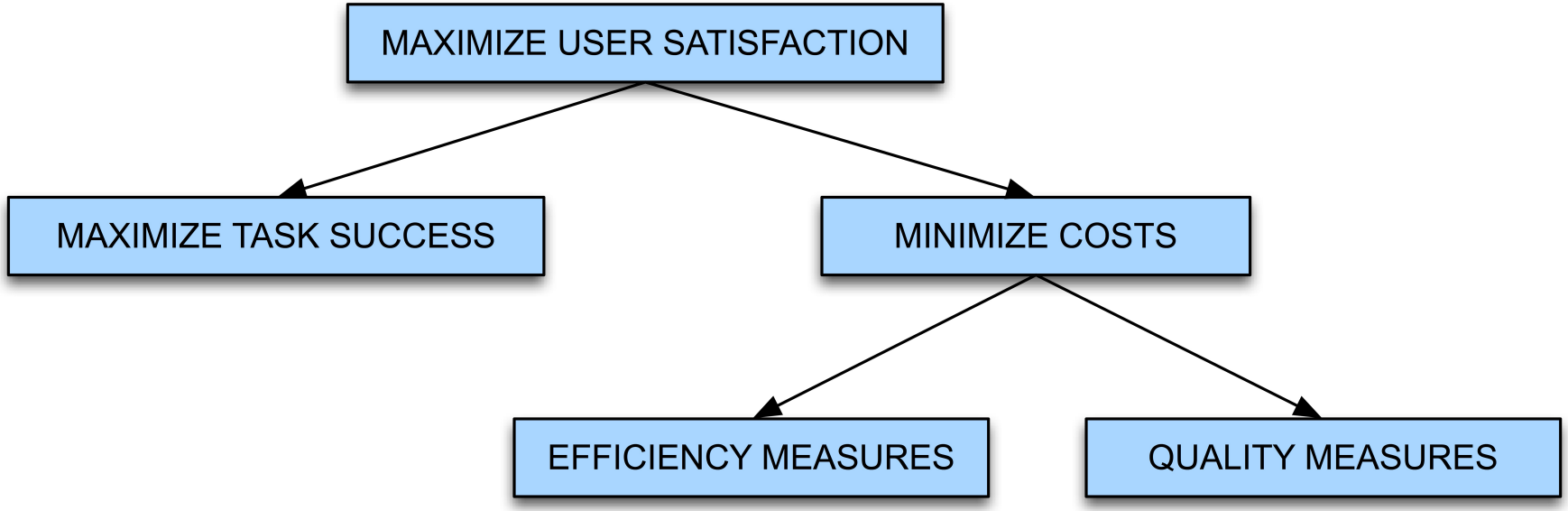
- “Concept accuracy” or “Concept error rate”
- % of semantic concepts that the NLU component returns correctly
- I want to arrive in Austin at 5:00
  - DESTCITY: Boston
  - Time: 5:00
- Concept accuracy = 50%
- Average this across entire dialogue
- “How many of the sentences did the system understand correctly”
- Can be used as either quality cost or task success

User Satisfaction: Sum of Many Measures

- Was the system easy to understand? (TTS Performance)
- Did the system understand what you said? (ASR Performance)
- Was it easy to find the message/plane/train you wanted? (Task Ease)
- Was the pace of interaction with the system appropriate? (Interaction Pace)
- Did you know what you could say at each point of the dialog? (User Expertise)
- How often was the system sluggish and slow to reply to you? (System Response)
- Did the system work the way you expected it to in this conversation? (Expected Behavior)
- Do you think you'd use the system regularly in the future? (Future Use)

PARADISE evaluation

- Maximize Task Success
- Minimize Costs
  - Efficiency Measures
  - Quality Measures
- PARADISE (PARAdigm for Dialogue System Evaluation) (Walker et al. 2000)



Evaluation Summary

- Best predictors of User Satisfaction:
  - Perceived task completion
  - mean recognition score (concept accuracy)
- Performance model useful for system development
  - Making predictions about system modifications
  - Distinguishing ‘good’ dialogues from ‘bad’ dialogues
  - As part of a learning model