CS 224S / Linguist 285
Spoken Language Processing

Andrew Maas | Stanford University | Spring 2024

Lecture 5: Course Project Intro and Q&A

Outline

- Course project introduction
- Project suggestions

Course Project Introduction

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**

Learning goals for this course

- **You will develop expertise on working with spoken language using modern tools, and create a foundation for contributing to spoken language system development and research**
- **Questions you should be able to answer after this course**
  - How are the data and use cases for spoken language different from written language NLP, computer vision, or robotics tasks?
  - What are modern tools you might use in industry or research to develop a fully capable spoken language application? (e.g. for speech recognition, synthesis, voice cloning, and dialog tasks)
  - What is the most recent research on deep learning models and approaches to spoken language tasks? How do they compare with deep learning architectures for other domains?
  - For a new product or research project, what process and tools would you pursue to effectively work with spoken language? Which tools might you need to modify vs use as-is?

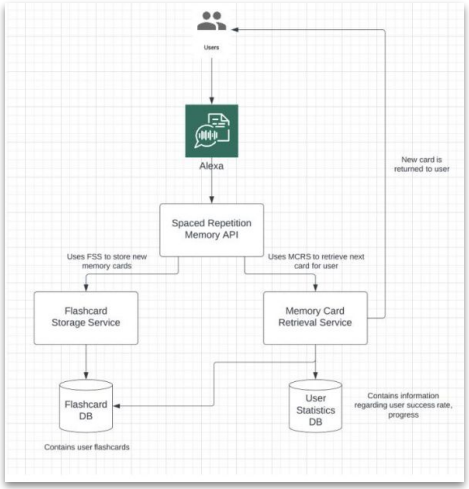
A Successful Project

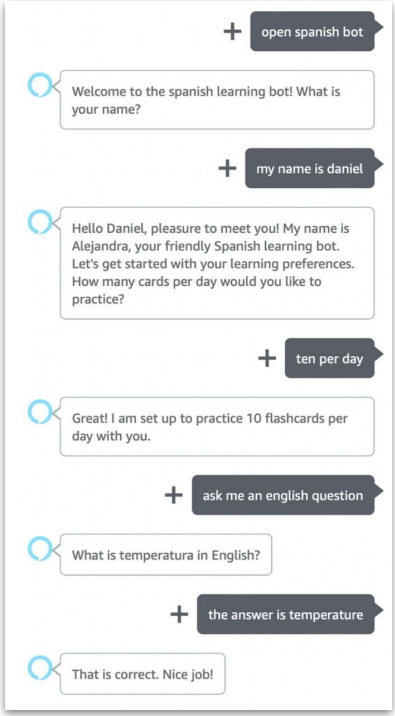
- Course-relevant topic. Proposed experiments or system address a challenging, unsolved SLP problem
- Proposes and executes a sensible approach informed by previous related work
- Performs error analysis to understand what aspects of the system are good/bad
- Adapts system or introduces new hypotheses/components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

Sample Projects

Language Learning through Spaced Repetition Memorization with Amazon Alexa

Candice Penelton Gurdeep Sullan Daniel Zhang





Inclusive ASR: Extending Transcription Systems for Stuttered Speech

German Enik Hannah Zhang

Spoken Language Diarization for Low Resource Languages via Few-Shot Transfer

Yubo Han Xuan Su Yian Zhang

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)
    - Response generation and speech synthesizer for spoken responses
- 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

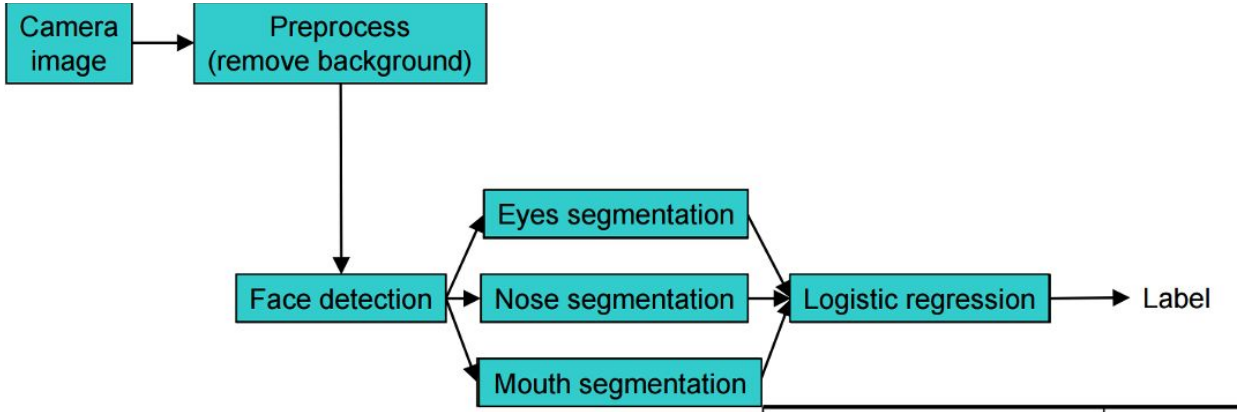
Balancing Scale and Depth

- **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 can 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%

Figure: Andrew Ng's CS229 lecture on applying ML [Lecture]

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?

From: Andrew Ng's CS229 lecture on applying ML Lecture

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.

From: Andrew Ng's CS229 lecture on applying ML Lecture

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?**
- **Model size and model training**
  - Large deep learning models are exciting! And require a lot of data + compute
  - Fine tuning or using foundation models can work well.
  - Ensure there is a model available for fine tuning or direct use (as part of proposal ideally)
- **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?
  - **In-class check in:** 2 minute update on progress, early learnings, and next steps
  - **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
  - We encourage you to include links to audio samples for TTS. Screen capture videos for dialog interactions as part of your final report
  - Much easier to understand your contribution this way than leave us to guess.
    Even if it doesn't quite work
  - Use laptop / phone to demo at poster session if you like

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 recognition/synthesis APIs, LLMs, Alexa, Kaldi, Whisper, etc.
- Use tools to focus on your project hypotheses
- Cite all tools you use. Cite code generation / copilot tools if you use them

Error Analysis with Tools

- **Project write up / 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)**

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

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

Table: 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. (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
- Great to explore LLMs, but try to focus on dialog systems that
    - Are task-oriented (actually take actions / require grounding in the world)
    - Use spoken language. Helps select a task and confront UX challenges in spoken vs written

Speech Recognition

- **Standard setup for experiments:**
    - 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. Explore foundation models + multilingual systems**
- **Successful projects do not need to improve on best existing results**
- **Adapting pre-trained neural approaches to new domains/tasks is great!**
    - Homework 2 provides a whisper fine tuning recipe you can build upon

Speech Synthesis

- **New set of problems to solve now that synthesis is good!**
    - Voice cloning and “designing” a voice
    - Controlling prosody and accent in fine-grained ways
    - Inferring correct prosody from text for more emotive voices
    - Managing multi-speaker TTS models and understanding their speaker embedding spaces
- **Evaluation is difficult. No single metric**
    - **Blizzard challenge provides training data and systems for comparison**
- **Matching state of the art can be very tedious signal processing or data quality issues**
- **You can do meaningful research without having a perfect TTS system in all regards (e.g. okay to have lower voice quality if your main focus is prosody)**

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 architectures
- 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
- Fine tuning whisper is a widely used recipe for ASR
- Many public models to use as a starting point. Your project should ask a research question or build a system that goes beyond simply “using a public model for its intended purpose”

Summary

- Have fun
- Build something you're proud of
- Post on Ed or use Andrew + Tolúlọpẹ́ office hours for early project feedback
- Proposals due April 24
- Reach out to project suggestors now/soon
- Post on Ed to find shared interests for project groups

Some Basic Ethics When Working on Speech Technologies

Don’t record someone without their consent

In California, all parties to any confidential conversation must give their consent to be recorded. For calls occurring over cellular or cordless phones, all parties must consent before a person can record, regardless of confidentiality.

Don’t create a speech synthesizer / voice clone of someone without their consent

It might be fun, but it’s a little creepy. People get upset.

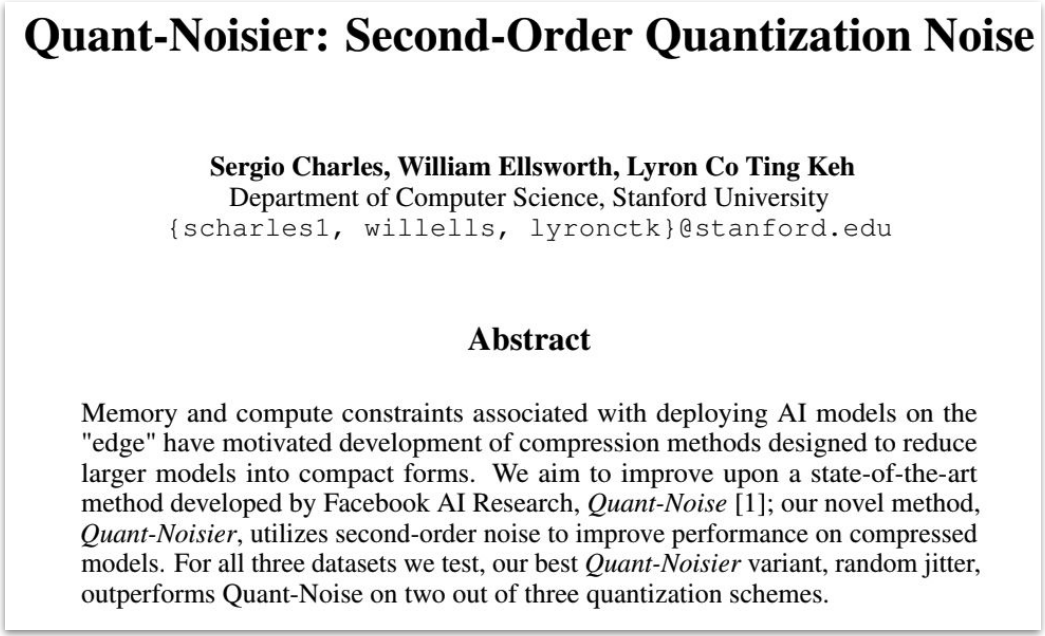
Okay to use existing speech datasets (we’ll provide some).

Do consider subgroup and language bias when building systems

Poor performance on subgroups e.g. non-native speakers

Many languages are under-served relative to English/Mandarin.

More Sample Projects

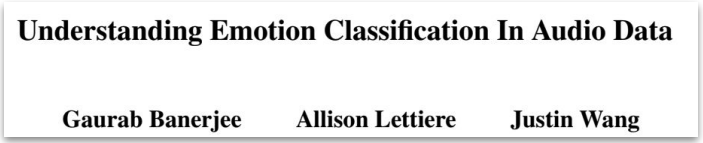


Quant-Noisier: Second-Order Quantization Noise

Sergio Charles, William Ellsworth, Lyron Co Ting Keh
Department of Computer Science, Stanford University
{scharles1, willells, lyronctk}@stanford.edu

Abstract

Memory and compute constraints associated with deploying AI models on the "edge" have motivated development of compression methods designed to reduce larger models into compact forms. We aim to improve upon a state-of-the-art method developed by Facebook AI Research, *Quant-Noise* [1]; our novel method, *Quant-Noisier*, utilizes second-order noise to improve performance on compressed models. For all three datasets we test, our best *Quant-Noisier* variant, random jitter, outperforms Quant-Noise on two out of three quantization schemes.

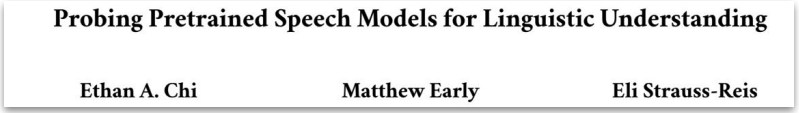


Understanding Emotion Classification In Audio Data

Gaurab Banerjee **Allison Lettiere** **Justin Wang**

Stock Return Prediction with Earnings Call Audio

Nick Steele, Stanford University, 450 Serra Mall, Stanford, CA 94305, nsteele@stanford.edu



Probing Pretrained Speech Models for Linguistic Understanding

Ethan A. Chi **Matthew Early** **Eli Strauss-Reis**

Previous projects: https://web.stanford.edu/class/cs224s/semesters/2024-spring/project