Georgetown University

ANLY-580
Natural Language Processing

Fall 2021
Lecture 1
Instructor: Chris Larson

August 25, 2021

Lecture 1

- Course logistics
- Course overview (30 min)
- Math refresher (100 min)
  - Linear algebra
  - Probability theory
  - Neural networks
- Lab (30 min):
  - Setup python3 environment
  - Setup Github
  - Bash tutorial (optional)

Course logistics

- Instructors:
  - Chris Larson (chris.larson@georgetown.edu)
- Teaching Assistants:
  - Yunfei Zhang (yz678)
  - Faye Wang (yw708)
- Section 1:
  - Lecture: Mon 6:30-9pm
  - Location: Remote
- Section 2:
  - Lecture: Thu 3:30-6pm
  - Location: Intercultural Center Rm 115
- Office hours:
  - Wed 4-6pm (Both sections)
  - Location: TBD
  - Start next week

More course logistics

- Prerequisites:
  - ANLY-501 Machine Learning
  - Working knowledge of Python3 + numerical/DL packages, git/Github
- Course syllabus is on Github
- Course texts:
  - Jurafsky, Martin. *Speech and Language Processing (3rd ed. draft)*
  - Eisenstein. *Natural Language Processing*
- Lecture notes
  - Lecture slides will be uploaded to Canvas ahead of each lecture
  - Course materials also posted here: https://github.com/chrislarson1/GU-ANLY-580-FALL-2021.git
  - To get access, you MUST email me with your GitHub user handle and I will invite you
- Expectations
  - Academic integrity & collaboration policy (handout on Canvas)
  - For section 2ers, please show up to lectures in person!
  - Laptops closed during lectures. And please, no social media.

Course grade

Component	Weight	Description
Entrance Exam	5%	Take home, individual
Placement Exam	10%	Take home, groups <= 4
Assignment 1	10%	individual
Assignment 2	10%	individual
Assignment 3	10%	individual
Final Project	50%	Groups <= 4
Lab participation	5%	--

Entrance exams

- Entrance Exam 1:
  - Covers course expectations & policies
  - Posted on Canvas/Github
  - **Due date**: Aug 29 (Section 1) / Sep 1 (Section 2)

- Entrance Exam 2:
  - Technical exam
    - Can complete in groups of <= 3
    - Take home, internet use encouraged
    - If it’s easy, great .. if it’s hard, that’s Ok. Goal is everyone starts from level set
  - Posted on Github
  - **Due date**: Sep 8 (Section 1) / Sep 12 (Section 2)

What will you learn?

- Birds eye view:
  - How to represent written and spoken language in a useful way
  - How to frame language understanding as a tractable statistical inference problem
  - How to evaluate the performance of language and speech processing systems
  - NLP systems building experience

- Canopy view:
  - State-of-the-art language modeling methods
  - Common NLP tasks (e.g., NER, QA) and their relation to various technologies (e.g., chatbots)
  - A perspective of how NLP has evolved and where it's headed
  - You will learn to use powerful tools within the NLP ecosystem
    - DL packages (e.g., Pytorch, Tensorflow, MXNet)
    - The *transformers* library
    - GPU computing

Georgetown University

About this course

Classic NLU debate: What does it mean to *understand*?

- Determine truth value?
- Determine entailment?
- Translate into another language?
- Respond in an appropriate way?

Prevailing perspectives

James Allen [1987]

There can be two underlying motivations for building a computational theory. The technological goal is simply to build better computers, and any solution that works would be acceptable. The cognitive goal is to build a computational analog of the human-language-processing mechanism; such a theory would be acceptable only after it had been verified by experiment.

Noam Chomsky [1996]

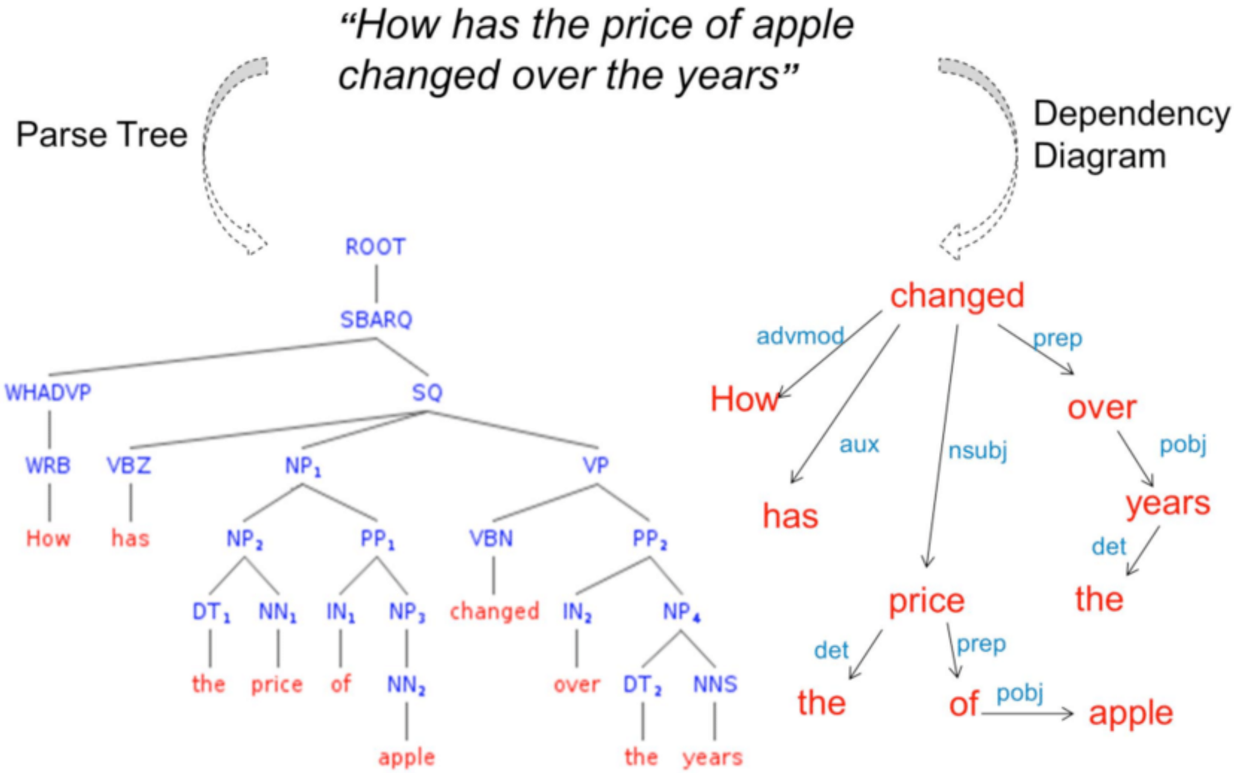
The question of whether a computer is playing chess, or doing long division, or translating Chinese, is like the question of whether robots can murder or airplanes can fly — or people; after all, the "flight" of the Olympic long jump champion is only an order of magnitude short of that of the chicken champion (so I'm told). These are questions of decision, not fact; decision as to whether to adopt a certain metaphoric extension of common usage.

NLU pre 2010

- **Based on syntactic parsing**
  - Maps text to a structured representation, or logical form, over which rules-based and/or probabilistic reasoning can be conducted to achieve some task
  - Relies on POS tagging, which are annotations of words in a sentence with their syntactic roles
  - Not robust to variations in raw input!

Notable successes

- Eliza (1966)
- IBM Watson (2011)



Parts of speech (POS)

Noun
Verb
Pronoun
Adverb
Adjective
Conjunction
Preposition
Interjection

Tag	Description	Example	Tag	Description	Example
CC	coordin. conjunction	*and, but, or*	SYM	symbol	+,%, &
CD	cardinal number	*one, two*	TO	"to"	*to*
DT	determiner	*a, the*	UH	interjection	*ah, oops*
EX	existential 'there'	*there*	VB	verb base form	*eat*
FW	foreign word	*mea culpa*	VBD	verb past tense	*ate*
IN	preposition/sub-conj	*of, in, by*	VBG	verb gerund	*eating*
JJ	adjective	*yellow*	VBN	verb past participle	*eaten*
JJR	adj., comparative	*bigger*	VBP	verb non-3sg pres	*eat*
JJS	adj., superlative	*wildest*	VBZ	verb 3sg pres	*eats*
LS	list item marker	*1, 2, One*	WDT	wh-determiner	*which, that*
MD	modal	*can, should*	WP	wh-pronoun	*what, who*
NN	noun, sing. or mass	*llama*	WP$	possessive wh-	*whose*
NNS	noun, plural	*llamas*	WRB	wh-adverb	*how, where*
NNP	proper noun, sing.	*IBM*	$	dollar sign	*$*
NNPS	proper noun, plural	*Carolinas*	#	pound sign	*#*
PDT	predeterminer	*all, both*	"	left quote	' or "
POS	possessive ending	*'s*	"	right quote	' or "
PRP	personal pronoun	*I, you, he*	(	left parenthesis	[, (, {, <
PRP$	possessive pronoun	*your, one's*	)	right parenthesis	],), }, >
RB	adverb	*quickly, never*	,	comma	,
RBR	adverb, comparative	*faster*	.	sentence-final punc	. ! ?
RBS	adverb, superlative	*fastest*	:	mid-sentence punc	: ; ... – -
RP	particle	*up, off*			

Figure 10.1 Penn Treebank part-of-speech tags (including punctuation).

, prestige
ecome
e, they
extremely
er
, while, whereas
into
, alas

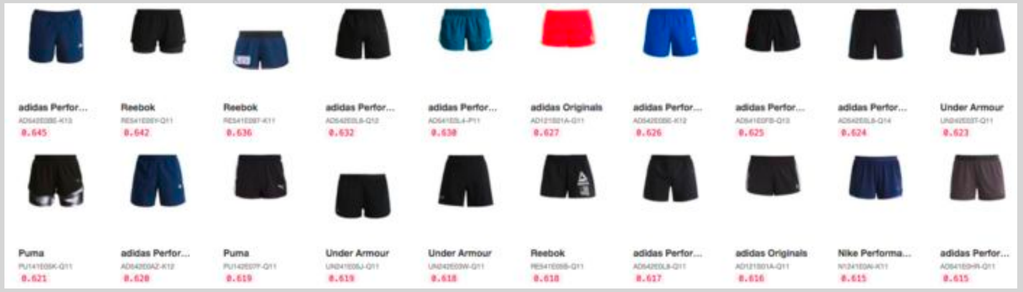
NLU post 2010, meet the Muppets

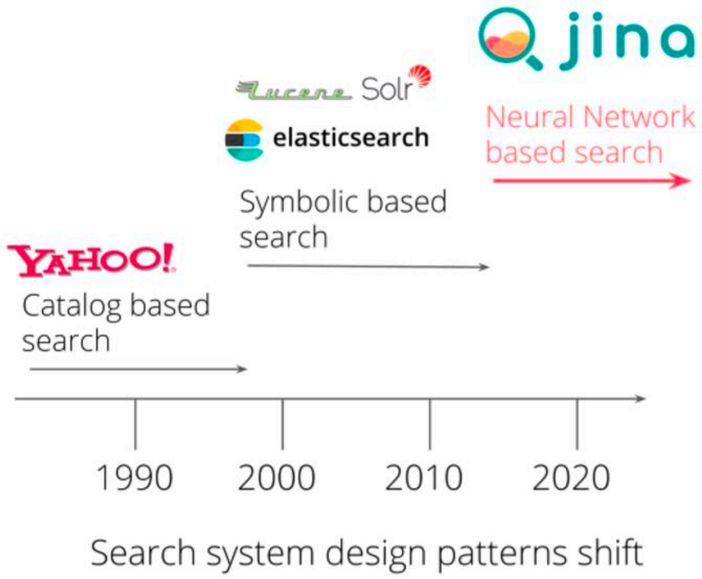
- **End-to-end modeling with no explicit syntactic representation**
  - Neural networks pertained on very large text corpora where goal is to predict words given then context (i.e., surrounding words).
  - Transfer / few shot learning to fine tune on downstream tasks
  - **This course primarily focuses on neural-based approaches**

Application areas: Web search

- Based on neural information retrieval techniques
- Improved semantic relevance
- Robust to variation in spelling, phrasing etc.
- Drastic improvement in image search relevance

query: *addisads trousers”*
result:



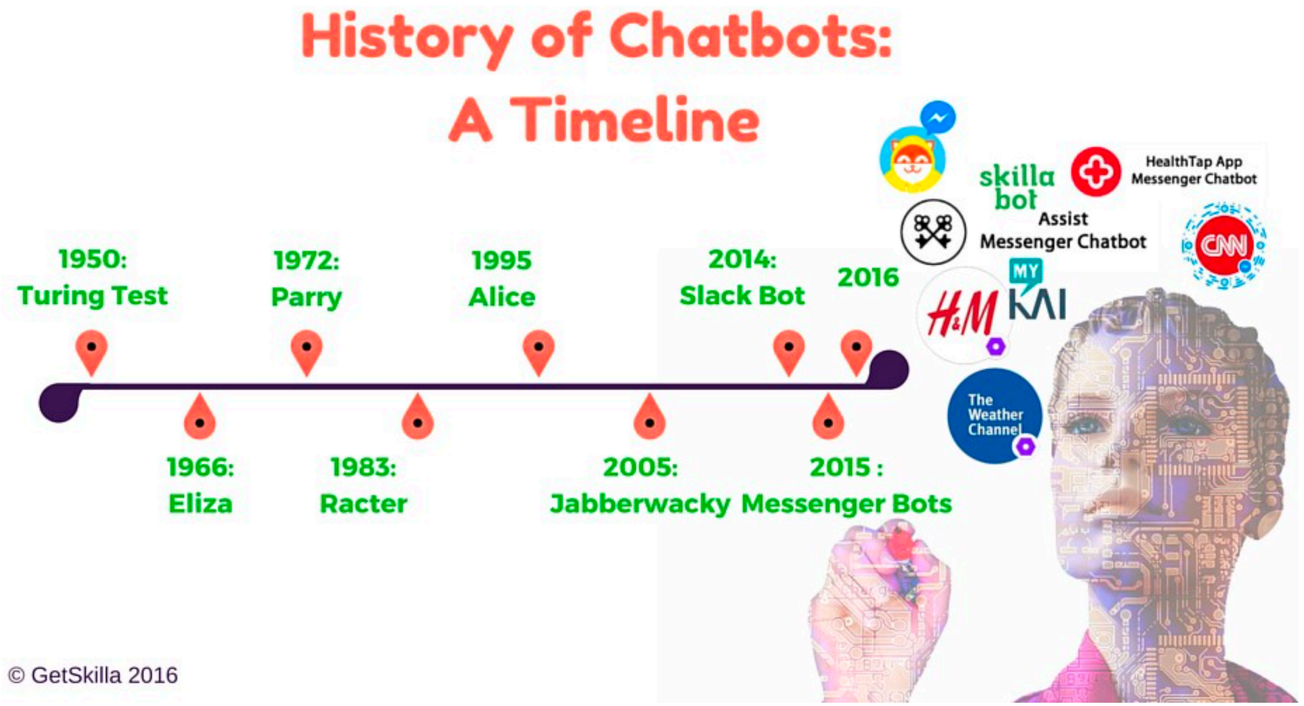


Search system design patterns shift

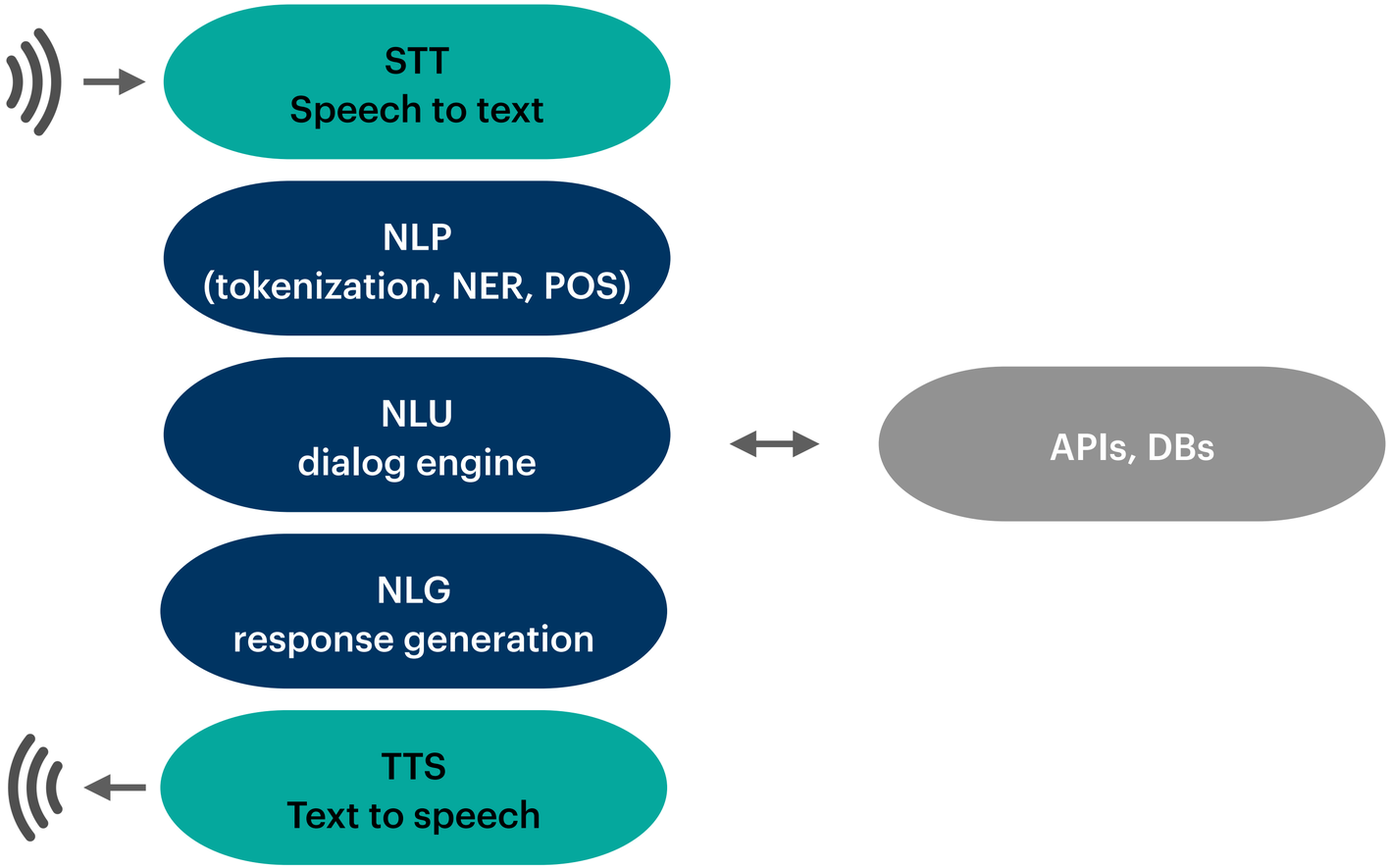
Application areas: Language translation

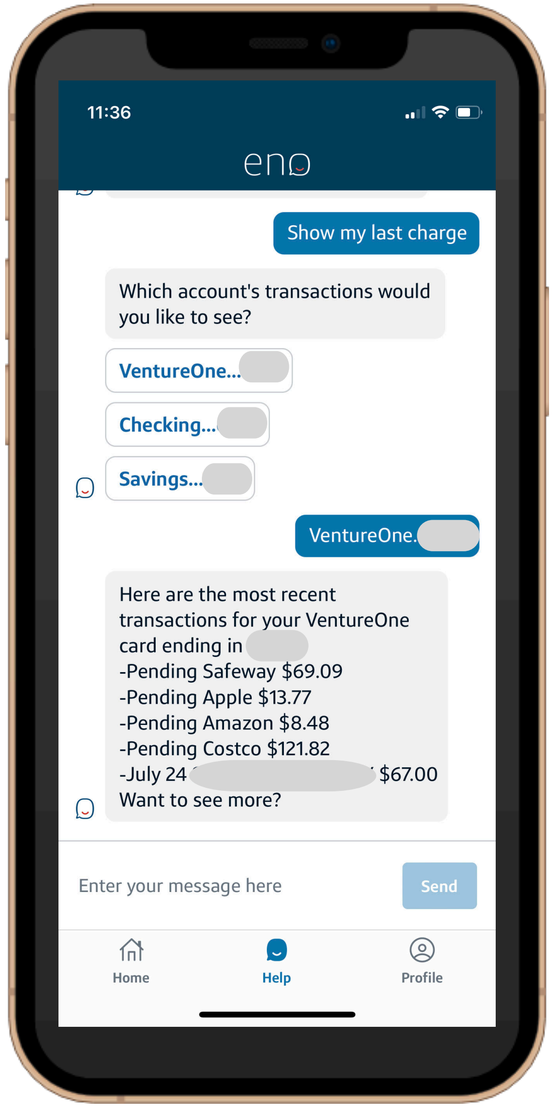
- Originally based on seq2seq models
- Now uses transformer architectures

Intelligent chatbots aren’t quite here yet



Application areas: Conversational bots



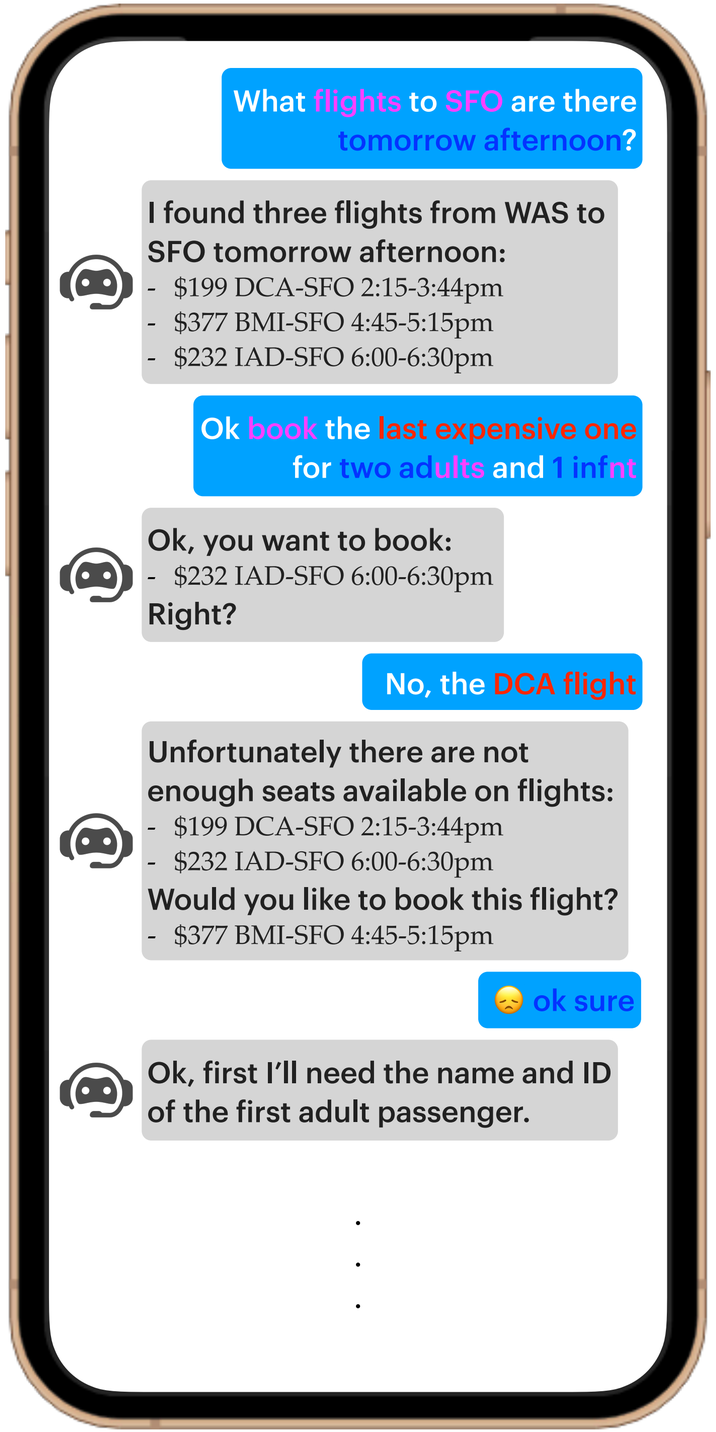


Why is conversation is hard?

World Knowledge

Domain knowledge

Discourse knowledge



Successes and shortcomings of current NLP systems

Successes

- Web search is now conversational, relevance has improved, remarkable long tail coverage.
- Machine translation is (essentially) a solved problem
- STT word error rates are low enough to make voice systems broadly useful in the real world
- TTS systems are able to generate realistic sounding voices
- Text generation systems (GPT family of models), within certain confined settings, pass writer's Turing test.
- Voice/chat bots have fundamentally changed HCI

Shortcomings

- AI assistants fall far short of human level interaction. At the highest level, this might/probably suggest(s) that learning textual representations alone is not sufficient to achieve this goal. In some ways this seems obvious. The problems of reference resolution and broad domain knowledge seem to be primary culprits, though this is a small slice of the problems that exist. **NLP is an exciting field to be in!**
- AIs built on logical representations of text are naturally equipped to perform logic & arithmetic, e2e AIs built on DNNs are not.
- Learning from voice/text, not surprisingly, means learning biases in those data; building systems around those biases propagates those biases. Technology should be for the benefit of society, thus fair and inclusive; thus making systems fair and inclusive is an important area of R&D.

Georgetown University

Refresher: Linear Algebra

First, why is this important?

By convention, we typically represent data as a matrix of values

- Columns correspond to features
- Rows correspond to observations of those features

$$D = \begin{bmatrix} x_{1,1} & \cdots & x_{1,N} \\ \vdots & \ddots & \vdots \\ x_{M,1} & \cdots & x_{M,N} \end{bmatrix}$$

where $x_{i,j}$ represents j^{th} feature value of i^{th} observation

N = number of features

M = number of observations

- Linear algebra underpins much of language modeling with neural networks, topic modeling, LSA, etc...

Vector spaces, inner & outer products

- In this class we represent data in a vector space
  - This means a data point lies in a grid (2D), cube (3D), or hypercube (4D+)

  ... in the general case it's *N*-dimensional
  - A point in space is represented by a vector
    - vector: $\mathbf{x} \in \mathbb{R}^N$
    - its scalar components: $x_i \in \mathbb{R} \quad where \quad 0 \leq i < N$
    - concretely: $\mathbf{x} = [x_0, x_1, \ldots, x_{N-1}]^T$

- Inner product (dot product in this class): $\langle \mathbf{a}, \mathbf{b} \rangle = \sum_{i=0}^{N-1} a_i \, b_i \quad where \quad \mathbf{a}, \mathbf{b} \in \mathbb{R}^N$

- Outer product: $\mathbf{a} \otimes \mathbf{b} = \mathbf{a}\mathbf{b}^T = \begin{bmatrix} a_1 b_1 & \cdots & a_1 b_N \\ \vdots & \ddots & \vdots \\ a_N b_1 & \cdots & a_N b_N \end{bmatrix}$

Tensors

- Formal definition: context dependent
- Definition in this class: A linear transformation within or between vector spaces
- Another definition: $$\mathbf{Ax} = \begin{bmatrix} A_{1,1} & \cdots & A_{1,N} \\ \vdots & \ddots & \vdots \\ A_{N,1} & \cdots & A_{N,N} \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ \vdots \\ x_N \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^{N} A_{1,i}\ x_i \\ \vdots \\ \sum_{i=1}^{N} A_{N,i}\ x_i \end{bmatrix}$$
- Matrix product (general form of op above): $$\mathbf{AB} = \sum_{k=0}^{N-1} A_{i,k} B_{k,j}$$
- Hadamard (a.k.a. element-wise, a.k.a. Schur) product: $$\left(\mathbf{A} \circ \mathbf{B}\right)_{i,j} = (A_{i,j})(B_{i,j})$$

$$\mathbf{A} \circ \mathbf{B} = \begin{bmatrix} A_{1,1} & \cdots & A_{1,N} \\ \vdots & \ddots & \vdots \\ A_{N,1} & \cdots & A_{N,N} \end{bmatrix} \cdot \begin{bmatrix} B_{1,1} & \cdots & B_{1,N} \\ \vdots & \ddots & \vdots \\ B_{N,1} & \cdots & B_{N,N} \end{bmatrix} = \begin{bmatrix} A_{1,1}B_{1,1} & \cdots & A_{1,N}B_{1,N} \\ \vdots & \ddots & \vdots \\ A_{N,1}B_{N,1} & \cdots & A_{N,N}B_{N,N} \end{bmatrix}$$

Vector and tensor norms

- Measure of the size, or magnitude of a vector or tensor

- Definition: The following criteria qualify $f(\cdot)$ as a norm:
  - Positive definite: $f(\mathbf{x}) = 0 \quad iff \quad \mathbf{x} = \mathbf{0}$
  - Triangle inequality: $f(\mathbf{x} + \mathbf{y}) \leq f(\mathbf{x}) + f(\mathbf{y})$
  - Homogeneity: $\forall \alpha \in \mathbb{R} : f(\alpha \mathbf{x}) = |\alpha| f(\mathbf{x})$

- The ones we care about are:

  - Lp norm: $\|\mathbf{x}\|_p = \left(\sum_i |x_i|^p \right)^{1/p}$

  - Frobenius norm: $\|\mathbf{A}\|_F = \sqrt{\sum_{i,j} A_{i,j}^2}$

Distance metrics

- The two distance metrics used most often in machine learning are the Manhattan (L1) and Euclidean (L2) distances, which can be defined using the Lp norm:

  - Manhattan: $\|\mathbf{x_1} - \mathbf{x_2}\|_1 = \sum_i |x_i^{(1)} - x_i^{(2)}|$

  - Euclidean: $\|\mathbf{x_1} - \mathbf{x_2}\|_2 = \sqrt{\sum_i (x_i^{(1)} - x_i^{(2)})^2}$

Matrix rank

- A full rank matrix, $\mathbf{A} \in \mathbb{R}^{N \times N}$, is one in which has *N* linearly independent column vectors. Two key implications arise from this:
  - Transforming the vectors space spanning $\mathbb{R}^N$, the resultant set also spans $\mathbb{R}^N$. So we say that the matrix $\mathbf{A}$ spans $\mathbb{R}^N$.
  - A logical extension is that ***A*** is a unique mapping, i.e. $\mathbf{Ax} = \mathbf{y}$ has a unique solution, ***x***, for all ***y*** $\in \mathbb{R}^N$.
- Matrices with rank < N are referred to as ***singular***.

Singular matrix?

$$\begin{bmatrix} A_{1,1} & A_{1,2} & A_{1,3} \\ A_{2,1} & A_{2,2} & A_{2,3} \\ A_{3,1} & A_{3,2} & A_{3,3} \end{bmatrix} = \begin{bmatrix} 1 & 2 & 7 \\ 0.5 & 5 & 15.5 \\ 1 & 3 & 10 \end{bmatrix}$$

Singular. 3rd col is linear combination of first two

$$1A_{:,1} + 3A_{:,2} = A_{:,3}$$

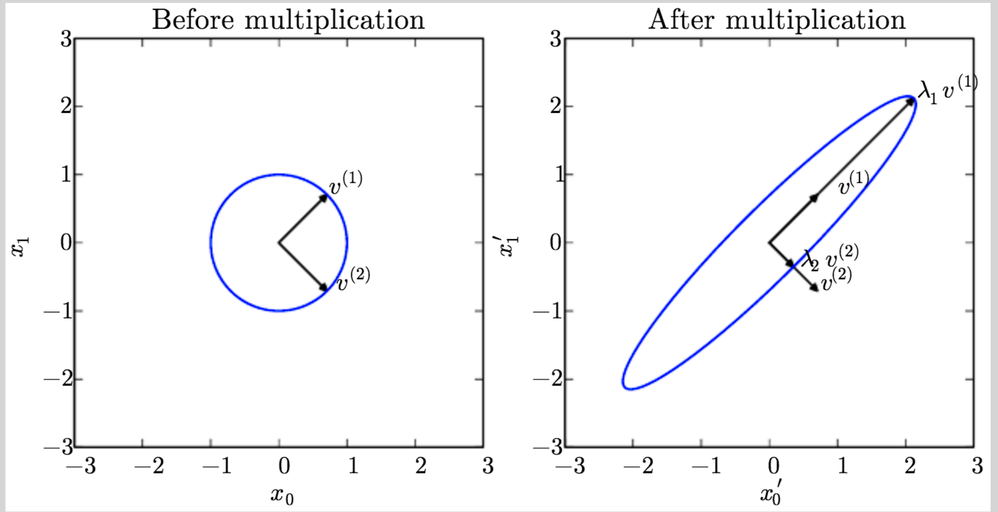
Singular matrix?

$$\begin{bmatrix} A_{1,1} & A_{1,2} & A_{1,3} \\ A_{2,1} & A_{2,2} & A_{2,3} \\ A_{3,1} & A_{3,2} & A_{3,3} \end{bmatrix} = \begin{bmatrix} 1 & 6 & 7 \\ 0.5 & 5 & 15.5 \\ 1 & 3 & 10 \end{bmatrix}$$

Full rank, all columns are linearly independent

Eigendecomposition

- Definition: $\mathbf{A}\mathbf{v} = \lambda\mathbf{v}$
- Decomposition: $\mathbf{A} = \mathbf{Q}\boldsymbol{\Lambda}\mathbf{Q}^T$ *where* $\mathbf{Q} = \begin{bmatrix} v_1^{(1)} & \dots & v_1^{(N)} \\ \vdots & \vdots & \vdots \\ v_N^{(1)} & \dots & v_N^{(N)} \end{bmatrix}$ *and* $\boldsymbol{\Lambda} = \begin{bmatrix} \lambda_1 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & \lambda_N \end{bmatrix}$
- Applies only to square matrices



Goodfellow et al., 2016, *Deep Learning*

Singular value decomposition

- What if our matrix is not square? SVD is a widely used factorization method in this case.

- Definition: $\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T \quad where \quad \mathbf{A} \in \mathbb{R}^{M \times N}, \quad \mathbf{U} \in \mathbb{R}^{M \times M}, \quad \mathbf{\Sigma} \in \mathbb{R}^{M \times N}, \quad \mathbf{V} \in \mathbb{R}^{N \times N}$

$$\mathbf{A} = \begin{bmatrix} u_1^{(1)} & \dots & u_1^{(M)} \\ \vdots & \vdots & \vdots \\ u_M^{(1)} & \dots & u_M^{(M)} \end{bmatrix} \begin{bmatrix} \sigma_1 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & \sigma_N \\ \vdots & \vdots & \vdots \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} v_1^{(1)} & \dots & v_N^{(1)} \\ \vdots & \vdots & \vdots \\ v_1^{(N)} & \dots & v_N^{(N)} \end{bmatrix}$$

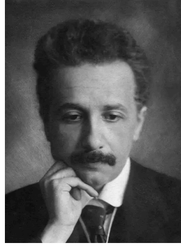
** M>N case*

- Numerically stable relative to eigendecomposition, widely used

Georgetown University

Refresher: Probability Theory

Why use randomness for randomness' sake?



"God does not play dice ..."
- Albert Einstein

- Is there a good justification for modeling deterministic processes as random ones? ...

- The argument of whether or not some process or phenomenon is truly random is often a philosophical matter. It is a matter of fact, though, that deterministic processes can, to those who only partially observe them, appear random.

- In the world of NLU, this is particularly relevant. Language is a coarse proxy of the human mind, and as such, building a deterministic model that mimics human perception and behavior is unrealistic. The process that generates human language is partially observable (text/speech); but we don't yet have a good model of human intelligence. Building a simple model that is uncertain about its belief is currently our best modeling approach.

- Dirty secret: Our decision to model $p(w \mid context)$ is also one of convenience: we have the internet and labels are free.

Random variables

- Examples: UV index, temperature, hair color, skin color, coin flip result etc..
- Probability distribution
  - Definition: How likely a random variable (or set of RVs) assumes each of its possible values. We call this a probability density function (pdf) for continuous RVs, and a probability mass function (pmf) for discrete valued RVs. In this class we are primarily concerned with PMFs.
  - A PMF, $P(x)$, must be:
    - Bounded: $\forall x \; in \; \mathrm{X} : 0 \leq P(x) \leq 1$
    - Normalized: $\sum_{x \in \mathrm{X}} P(x) = 1$

Gaussian distribution

- Univariate: $$N(x;\mu,\sigma^2) = (2\pi\sigma^2)^{-1/2} exp(-\frac{(x-\mu)^2}{2\sigma^2})$$

- Multivariate: $$N(\mathbf{x};\boldsymbol{\mu},\boldsymbol{\Sigma}) = \sqrt{\frac{1}{(2\pi)^N \det \boldsymbol{\Sigma}}} \exp(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})) \quad where \quad \mathbf{x} \in \mathbb{R}^N$$

Joint, marginal, conditional probabilities

- Joint: $P(x, y) \quad where \quad P(\cdot) \in \mathbb{R}^{|X| \times |Y|}$

- Marginal: $P(x) = \sum_{y} P(\mathsf{x} = x, \mathsf{y} = y)$

- Conditional: $P(y \mid x) = \frac{P(\mathsf{y} = y, \mathsf{x} = x)}{P(\mathsf{x} = x)}$

Product rule, independence, conditional independence

- Product rule: $P(x^{(1)}, \ldots, x^{(n)}) = P(x^{(1)}) \prod_{i=2}^{n} P(x^{(i)} \mid x^{(1)}, \ldots, x^{(i-1)})$

- Independence condition: $P(x, y) = P(x)P(y)$

- Conditional independence condition: $P(x, y \mid z) = P(x \mid z)\ P(y \mid z)$

Expected value and covariance functions

- Expectation: $\mathbb{E}_{x\sim P}[f(x)] = \sum_x P(x)f(x)$

- Variance: $Var(f(x)) = \mathbb{E}_x\left[\left(f(x) - \mathbb{E}[f(x)]\right)^2\right]$

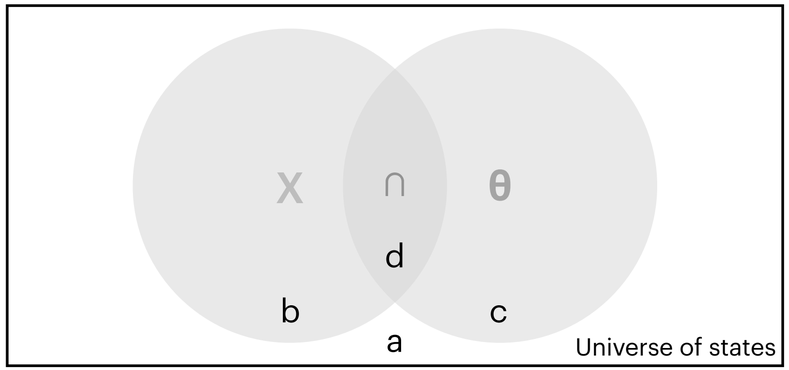
- Covariance: $Cov(f_1(x), f_x(x)) = \mathbb{E}\left[\left(f_1(x) - \mathbb{E}[f_1(x)]\right) \cdot \left(f_2(x) - \mathbb{E}[f_2(x)]\right)\right]$

- Covariance of random vector, ***x***: $$\begin{aligned} Cov(\mathbf{x}, \mathbf{x}) &= \mathbb{E}\left[(\mathbf{x} - \mathbb{E}[\mathbf{x}])(\mathbf{x} - \mathbb{E}[\mathbf{x}])^T\right] \\ &= \mathbb{E}\left[\mathbf{x}\mathbf{x}^T - \mathbb{E}[\mathbf{x}]\mathbb{E}[\mathbf{x}]^T\right] \end{aligned}$$

Bayes' Rule

- Intuitive statement: $P(x|\theta)P(\theta) = P(\theta|x)P(x)$

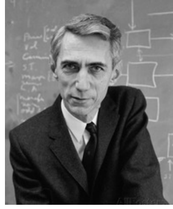
  - Proof $\frac{d}{c} \cdot \frac{c}{a} = \frac{d}{b} \cdot \frac{b}{a} = \frac{d}{a}$



- Bayes' Rule follows by deduction: $P(\theta|x) = \frac{P(x|\theta)P(\theta)}{P(x)}$ $posterior = \frac{likelihood \times prior}{evidence}$

- Prescription for how to update a model given new evidence (i.e., new data)

Similarity measures between distributions



Claude Shannon

- Shannon postulated that any measure of the informativeness of an event, x, should satisfy three conditions:
  1. An event with probability 1 yields no information
  2. The probability of an event and the information it yields vary inversely with each other
  3. The total information coming from independent events is purely additive

  - Which he used to define self-information: $I(x) = -\log P(x)$

- Shannon entropy: $H(P) = \mathbb{E}_{x\sim P}[I(x)] = -\mathbb{E}_{x\sim P}[\log P(x)]$

- Kullback-Leibler (KL) divergence: $D_{KL}(P||Q) = \mathbb{E}_{x\sim P}\left[\log \frac{P(x)}{Q(x)}\right]$

- Cross entropy: $H(P,Q) = H(P) + D_{KL}(P||Q) = -\mathbb{E}_{x\sim P}[\log Q(x)]$

Maximum likelihood estimation

- Estimates the parameters, $\boldsymbol{\theta}$, of a distribution using a likelihood function, $\mathscr{L}(\mathbf{D};\boldsymbol{\theta})$, given some data, ***D.***

- We maximize the likelihood function by minimizing its -logarithm:

$$\begin{aligned}\mathscr{L}(\boldsymbol{\theta}\,|\,\mathbf{D}) &= P(\mathbf{D};\boldsymbol{\theta}) \\ &= \left(\prod_{i=1}^{M} P(\mathbf{y}_i\,|\,\mathbf{x}_i;\boldsymbol{\theta})\right)^{\frac{1}{M}} \\ &= \frac{1}{M}\sum_{i=1}^{M} \log P(\mathbf{y}_i\,|\,\mathbf{x}_i;\boldsymbol{\theta})\end{aligned}$$

Note: technically natural log, but true to within a constant

$$\hat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}}{\operatorname{argmin}} \sum_{i=1}^{M} -\log P(\mathbf{y}_i\,|\,\mathbf{x}_i;\boldsymbol{\theta})$$

- This expresses an optimization problem; the form of $p(\mathbf{D};\boldsymbol{\theta})$ dictates how we solve it
  - In deep learning, this function is a neural network; we compute its gradient w.r.t. $\boldsymbol{\theta}$, and then estimate $\hat{\boldsymbol{\theta}}$ using stochastic gradient descent (SGD).

Georgetown University

Refresher: Neural Networks

Artificial neural networks

- Feedforward NN with one hidden layer:

$$\hat{\mathbf{y}} = \varphi(\mathbf{W}^{(2)}\sigma^{(1)}(\mathbf{W}^{(1)}\mathbf{x} + \mathbf{b}^{(1)}) + \mathbf{b}^{(2)}) = \varphi\left(\sum_{k=1}^{K} W_{kl}^{(2)}\, \sigma^{(1)}\left(\sum_{j=1}^{J} W_{jk}^{(1)} x_j + b_j^{(1)} \right) + b_k^{(2)} \right)$$

$\mathbf{x}$ = input layer

$\hat{\mathbf{y}}$ = output prediction layer

$\boldsymbol{\theta}$ = parameters to estimate = $\{\mathbf{W}^{(1)}, \mathbf{b}^{(1)}, \mathbf{W}^{(2)}, \mathbf{b}^{(2)}\}$

$$\sigma(\mathbf{z}) = \begin{cases} \max(\mathbf{0}, \mathbf{z}) & \text{relu, defacto standard} \\ \left(1 + e^{-\mathbf{z}}\right)^{-1} & \text{sigmoid, old school} \\ \text{many} & \text{variations on these and others} \end{cases}$$

$$\varphi(\mathbf{z}) = \begin{cases} h\tan \mathbf{z} & \textit{regression} \\ \dfrac{e^{\mathbf{z}}}{\sum_{\mathbf{z}} e^{\mathbf{z}}} & \textit{classification} \end{cases}$$

NN parameter estimation for regression

- Model:

$$\mathbf{Y} = \hat{\mathbf{Y}} + \boldsymbol{\epsilon} = f(\mathbf{X};\boldsymbol{\theta}) + \boldsymbol{\epsilon} \qquad \textit{where}$$

$f(\mathbf{X};\boldsymbol{\theta})$ expresses our neural network

$$\boldsymbol{\epsilon} \sim N(\mathbf{Y} - f(\mathbf{X};\boldsymbol{\theta}), \boldsymbol{\Sigma})$$

$$= \sqrt{\frac{1}{(2\pi)^N \det \boldsymbol{\Sigma}}} e^{-\frac{1}{2}\left(\mathbf{X} - f(\mathbf{X};\boldsymbol{\theta})\right)^T \boldsymbol{\Sigma}^{-1} \left(\mathbf{X} - f(\mathbf{X};\boldsymbol{\theta})\right)}$$

$$:= P(\mathbf{Y} \mid \mathbf{X};\boldsymbol{\theta}) \leftarrow \mathscr{L}(\mathbf{D};\boldsymbol{\theta})$$

- Optimization:

$$\hat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}}{\operatorname{argmin}} \; -\log P(\mathbf{Y} \mid \mathbf{X};\boldsymbol{\theta})$$

$$= \underset{\boldsymbol{\theta}}{\operatorname{argmin}} \; -\log\left(\sqrt{\frac{1}{(2\pi)^N \det \boldsymbol{\Sigma}}} \exp\left[-\frac{1}{2}\left(\mathbf{X} - f(\mathbf{X};\boldsymbol{\theta})\right)^T \boldsymbol{\Sigma}^{-1}\left(\mathbf{X} - f(\mathbf{X};\boldsymbol{\theta})\right)\right]\right)$$

$$= \underset{\boldsymbol{\theta}}{\operatorname{argmin}} \; -\frac{1}{2}\log((2\pi)^N \det \boldsymbol{\Sigma}) - \frac{1}{2}\left(\mathbf{X} - f(\mathbf{X};\boldsymbol{\theta})\right)^T \boldsymbol{\Sigma}^{-1}\left(\mathbf{X} - f(\mathbf{X};\boldsymbol{\theta})\right)$$

$$= \underset{\boldsymbol{\theta}}{\operatorname{argmin}} \; -\left(\mathbf{X} - f(\mathbf{X};\boldsymbol{\theta})\right)^T \left(\mathbf{X} - f(\mathbf{X};\boldsymbol{\theta})\right)$$

$$= \underset{\boldsymbol{\theta}}{\operatorname{argmin}} \; -\sum_{i=1}^{M} \left(\mathbf{y}^{(i)} - f(\mathbf{x}^{(i)};\boldsymbol{\theta})\right)^T \left(\mathbf{y}^{(i)} - f(\mathbf{x}^{(i)};\boldsymbol{\theta})\right)$$

$$= \underset{\boldsymbol{\theta}}{\operatorname{argmin}} \; -\sum_{i=1}^{M}\sum_{j=1}^{N} \left(\mathbf{y}_j^{(i)} - f(\mathbf{x}^{(i)};\boldsymbol{\theta})_j\right)^2 \quad \leftarrow \text{least squares}$$

$\boldsymbol{\Sigma}$ is diagonal, strictly positive, independent of $\mathbf{x}$; it doesn't affect $\hat{\boldsymbol{\theta}}$

NN parameter estimation for classification

- Model: $P(\mathbf{y} \mid \mathbf{x}; \boldsymbol{\theta}) \quad where \quad \mathbf{x}, \mathbf{y} \sim P_D \,, \quad \mathbf{y} \in \{0,1\}^N$

  one hot encoding

- Optimization:

$$\begin{aligned}
\hat{\boldsymbol{\theta}} &= \underset{\boldsymbol{\theta}}{\operatorname{argmin}} \sum_{i=1}^{M} -\log P(\mathbf{y}_i \mid \mathbf{x}_i; \boldsymbol{\theta}) \\
&= \underset{\boldsymbol{\theta}}{\operatorname{argmin}} \; -\mathbb{E}_{\mathbf{x},\mathbf{y} \sim P_D}\left[\log P(\mathbf{y} \mid \mathbf{x}; \boldsymbol{\theta})\right] \\
&= \underset{\boldsymbol{\theta}}{\operatorname{argmin}} \; \mathbb{E}_{\mathbf{x},\mathbf{y} \sim P_D}\left[\log \frac{1}{P(\mathbf{y} \mid \mathbf{x}; \boldsymbol{\theta})}\right]
\end{aligned}$$

↓

Cross entropy between data distribution and the model output distribution, $H\left(P_D(\mathbf{y} \mid \mathbf{x}), P(\mathbf{y} \mid \mathbf{x}; \boldsymbol{\theta})\right)$. Note, this holds for arbitrary PMFs (softmax, Bernoulli, etc...). Because the labels, $\mathbf{y}$, are one hot, they have zero entropy, and thus in this setting minimizing the cross entropy is equivalent to minimizing the KL divergence, $D_{KL}\left(P_D(\mathbf{y} \mid \mathbf{x}), P(\mathbf{y} \mid \mathbf{x}; \boldsymbol{\theta})\right)$.