TTIC 31110
Speech Technologies

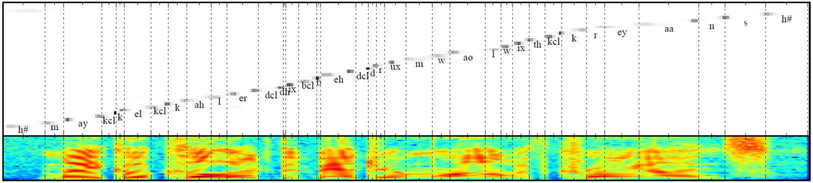
May 14, 2020

Announcements

- HW4 due Monday 5/25 7pm
- Revised project timeline for graduating seniors
- Project presentations
  - Will be split between June 2 and June 11
  - June 2 presentations will present work in progress rather than final results
  - We will be asking you to express a preference for one or the other

Question from last time: Visualizing attention

FDHC0_SX209: Michael colored the bedroom wall with crayons.



- Here decoder is producing phone labels
- Each decoder time step is one phone label
- Y-axis is decoder time steps, from bottom to top
- Each row is α_j for decoder time step j

Context vector computation :

$$\begin{aligned} \mathbf{c}_j &= \sum_{t=1}^{T} \alpha_{jt} \mathbf{h}_t \\ \alpha_j &= \mathrm{softmax}(\mathbf{u}_j) \\ u_{jt} &= \mathbf{h}_t^T \mathbf{s}_j \end{aligned}$$

Outline

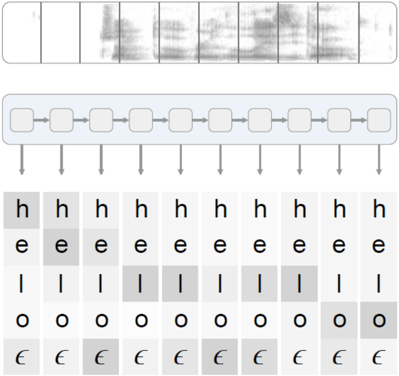
Connectionist temporal classification (CTC)

Language models

Connectionist temporal classification (CTC)

[Graves+ 2006]

From https://distill.pub/2017/ctc/:



RNN with softmax output layer produces a posterior probability for each label + ϵ

Basic ("greedy") CTC decoding

- RNN with softmax output layer produces a posterior probability for each label $+\ \epsilon$
- At each time frame, output the most likely frame label
- Finally, map frame labels to "collapsed" label sequence as follows:

h	h	e	ϵ	ϵ	l	l	l	ϵ	l	l	o

First, merge repeat characters.

h	e	ϵ	l	ϵ	l	o

Then, remove any ϵ tokens.

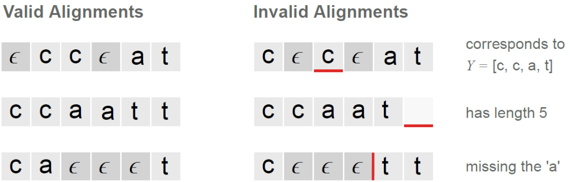
h	e		l		l	o

The remaining characters are the output.

h	e	l	l	o

CTC training

Given a sequence X of T acoustic frames and a corresponding label sequence Y with $L < T$ labels, e.g. the word **cat**, consider the set of all of the valid frame label sequences ("alignments") $\mathcal{A}_{X,Y}$



CTC training

Given a sequence X of T acoustic frames and a corresponding label sequence Y with $L < T$ labels, e.g. the word **cat**, consider the set of all of the valid frame label sequences ("alignments") $\mathcal{A}_{X,Y}$.

Then the CTC loss is a *marginal log loss*:

$$-\log p(Y|X) = -\log \sum_{A \in \mathcal{A}_{X,Y}} \prod_{t=1}^{T} p(a_t|X)$$

where $p(a_t|X)$ is the softmax output of the RNN at frame t

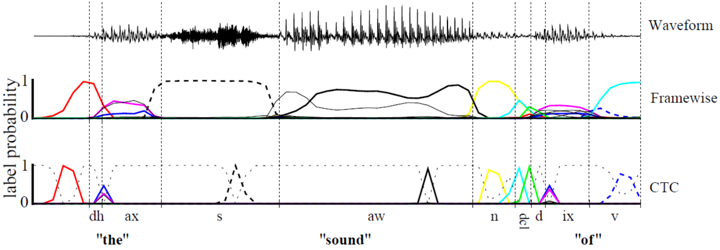
CTC training

CTC loss:
$-\log p(Y|X) = -\log \sum_{A \in \mathcal{A}_{X,Y}} \prod_{t=1}^{T} p(a_t|X)$

Looks hard to backprop, but it turns out to be equivalent to a forward-backward-like algorithm!

CTC posterior visualization

CTC posteriors vs. posteriors from an RNN trained with frame-level log loss (e.g. for a hybrid HMM/NN):



[Graves+ 2006]

CTC not so different from HMMs...

Putting HMM into CTC-like notation ($A =$ state sequence):

$p(X) = \sum_{A \in \mathcal{A}} \prod_{t=1}^{T} p(\mathbf{x}_t | a_t) p(a_t | a_{t-1})$

Suppose transition probabilities are uniform:

$p(X) \propto \sum_{A \in \mathcal{A}} \prod_{t=1}^{T} p(\mathbf{x}_t | a_t)$

CTC not so different from HMMs...

$p(X) \propto \sum_{A \in \mathcal{A}} \prod_{t=1}^{T} p(\mathbf{x}_t | a_t)$

2 differences from CTC:

- $p(\mathbf{x}_t | a_t)$ vs. $p(a_t | \mathbf{x}_t)$
- Definition of $\mathcal{A}$

Rewrite using Bayes rule (as we did for hybrid models):

$p(X) \propto \sum_{A \in \mathcal{A}} \prod_{t=1}^{T} p(a_t | \mathbf{x}_t) / p(a_t)$

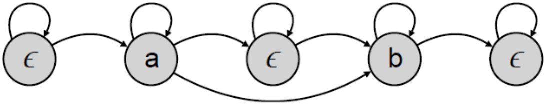
Assuming uniform priors of the labels (states):

$p(X) \propto \sum_{A \in \mathcal{A}} \prod_{t=1}^{T} p(a_t | \mathbf{x}_t)$

So computing the marginal probability in CTC is just like computing likelihood in HMMs, hence forward-backward algorithm!

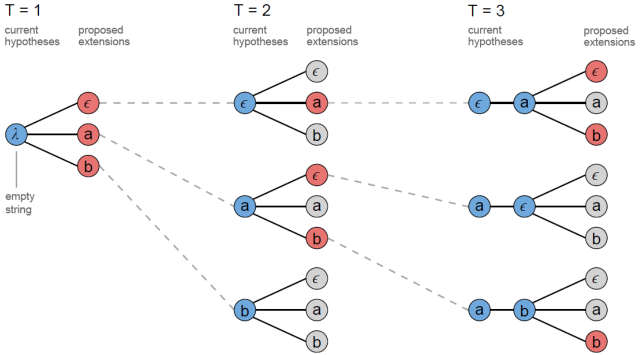
Equivalent HMM state diagram for CTC

Assuming the ground-truth sequence “a b”:



Back to decoding: Why not sum over alignments?

- Greedy decoding is similar to making the Viterbi approximation for HMMs
- Alternatively, could approximate the sum over all alignment paths corresponding to the same label sequence, with a beam search. (This is rarely done, though)



Connection between CTC and encoder-decoder models

- Encoder = all but last layer of the RNN
- Decoder = softmax + label collapsing function

Attention models and CTC with other neural architectures

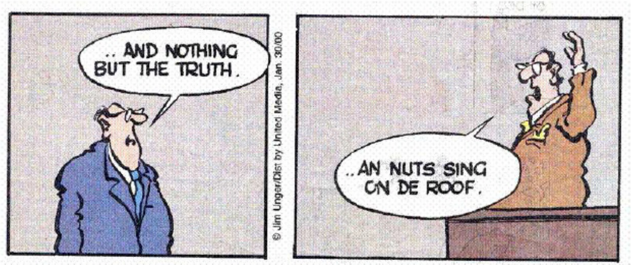
- We've discussed attention and CTC models in the context of RNNs
- But both can be used with other architectures
- E.g., convolutional or transformers

Reminder: The "fundamental equation of ASR"

$$
\begin{aligned}
\mathbf{w}^* &= \operatorname*{argmax}_{\mathbf{w}} p(\mathbf{w}|\mathbf{O}) \\
&= \operatorname*{argmax}_{\mathbf{w}} p(\mathbf{O}|\mathbf{w})p(\mathbf{w})
\end{aligned}
$$

- where $\mathbf{O}$ are the acoustic feature frames for an utterance, $\mathbf{w}$ is a word sequence
- $p(\mathbf{O}|\mathbf{w})$ is the acoustic model
- $p(\mathbf{w})$ is the language model
- Both are too complex to model directly; we factor them into manageable chunks

Why are language models needed?



Why are language models needed? (2)

- To disambiguate acoustically similar utterances using prior knowledge about word sequences
  - $p($"And nothing but the truth"$) = .0023$
  - $p($"And nuts sing on the roof"$) \approx 0$

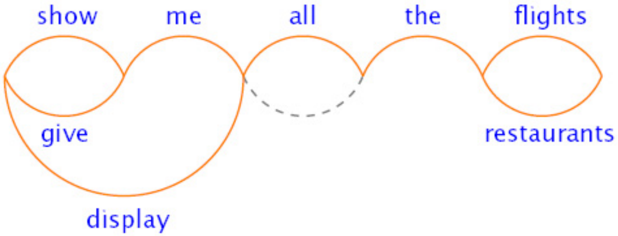
  - $p($"It is easy to recognize speech"$) = .0001$
  - $p($"It is easy to wreck a nice beach"$) = .00000001$

Language models are used in many applications

- Speech recognition
- Handwriting recognition
- Spelling correction
- Optical character recognition
- Machine translation
- Natural language generation
- ...
- Almost any problem that requires predicting a sequence

Deterministic LMs

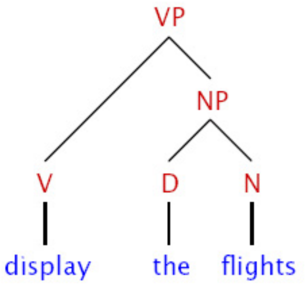
Finite-state grammars (finite-state automata/machines/networks)



- Allowable sequences are defined by a word graph
- Can also be described by *regular* rewrite rules, e.g. $A \rightarrow a, A \rightarrow aB$

Deterministic LMs (2)

Context-free grammars (CFGs)



- Allowable sequences are those that parse according to the grammar
- Can be described by *context-free* rewrite rules, e.g. $A \to a, A \to BC, A \to aA$

Deterministic LMs (3)

Word-pair grammars

- Enumerate all of the legal 2-word sequences in the language

show → me	me → all → the	the → flights → restaurants

- Was popular for some constrained tasks $\sim$ 3 decades ago

Deterministic vs. Statistical LMs

Deterministic LMs

- Can work well for simple menu-based tasks
- Typically have finite coverage $\longrightarrow$ possibly disastrous for ASR
- Don't distinguish between likely and unlikely sequences, only possible and impossible

Statistical LMs

- Assign a probability $p(\mathbf{w})$ to each word sequence $\mathbf{w} = (w_1, \ldots, w_K)$, subject to $\sum_{\mathbf{w}} p(\mathbf{w}) = 1$
- Probabilities are used to guide the search among alternative word hypotheses during recognition

History-based statistical LMs

- $p(\mathbf{w})$ can be expanded using the chain rule:

$$p(\mathbf{w}) = \prod_{i=1}^{K} p(w_i | w_1, \ldots, w_{i-1}) = \prod_{i=1}^{K} p(w_i | h_i)$$

  where $h_i = (w_1, \ldots, w_{i-1})$ is the *history* for word w_i.
- Note: First & last words typically assumed to be a sentence boundary marker, $w_1 = w_K = <>$
- Too many possible histories $\Rightarrow$ reduce to equivalence classes $\phi(h_i)$, such that $p(w_i|h_i) \approx p(w_i|\phi(h_i))$
- Good equivalence classes maximize the information about the current word given the class $\phi(h_i)$
- (LMs requiring the full word sequence $\mathbf{w}$ can be used, but usually in a "rescoring" setting – more later...)

n-gram language models

- In *n*-gram LMs, the history equivalence class is the previous $n-1$ words: $\phi(h_i) = (w_{i-1}, \ldots, w_{i-n+1})$
- For example:
  - bigram LM $p(w_i|w_{i-1})$
  - trigram LM $p(w_i|w_{i-1}, w_{i-2})$
- Trigrams were for a long time the dominant LM in large-vocabulary recognition research, but longer histories now being used with large training sets (even arbitrarily long histories)

n-gram example

Consider the sentence:
$\mathbf{w}$ = "*The quick brown fox jumped over the lazy dog.*"

$$
\begin{aligned}
p(w_1, \ldots, w_n) \quad = \quad & p(\text{the}| <>) \\
& p(\text{quick}|\text{the}, <>) \\
& p(\text{brown}|\text{quick, the}) \\
& \ldots \\
& p(\text{dog}|\text{lazy, the}) \\
& p(<> |\text{dog, lazy})
\end{aligned}
$$

Where do the probabilities come from?

Maximum-likelihood estimate of *n*-gram probabilities given some training set of text:

$$\hat{p}(\text{quick}|<>,\text{the}) = \frac{\text{count}(<>,\text{the, quick})}{\text{count}(<>,\text{the})}$$

Example of language model impact

Resource Management task

- Speaker-independent, continuous-speech corpus
- Sentences generated from a finite-state grammar
- 997-word vocabulary

	No LM	Word-Pair	Bigram
% Word Error Rate	29.4	6.3	4.2

Trigram example (Jelinek '97)

1	The	are	to	know	the	issues	necessary
2	This	will		have	this	problems	data
3	One	the		understand	these	the	information
4	Two	would		do	problems		above
5	A	also		get	any		other
6	Three	do		the	a		time
7	Please	need		use	problem		people
8	In			provide	them		operators
9	We			insert	all		tools
•				•			•
•				•			•
96				write			jobs
97				me			MVS
98				resolve			old
•							•
•							•
1639							reception
1640							shop
1641							important

Trigram example (2)

1	role	and	the	next	be	metting	of
2	thing	from			two	months	<>
3	that	in				years	
4	to	to				meetings	
5	contact	are				to	
6	parts	with				weeks	
7	point	were				days	
8	for	requiring					
9	issues	still					
•		•					
•		•					
61		being					
62		during					
63		I					
64		involved					
65		would					
66		within					

Random sentence generation example: Air travel domain bigram

Show me the flight earliest flight from Denver
How many flights that flight leaves around is the Eastern Denver
I want a first class
Show me a reservation the last flight from Baltimore for the first
I would like to fly from Dallas
I get from Pittsburgh
Which just small
In Denver on October
I would like to San Francisco
Is flight flying
What flights from Boston to San Francisco
How long can you book a hundred dollars
I would like to Denver to Boston and Boston
Make ground transportation is the cheapest
Are the next week on AA eleven ten
First class
How many airlines from Boston on May thirtieth
What is the city of three PM
What about twelve and Baltimore

Random sentence generation example: Air travel domain trigram

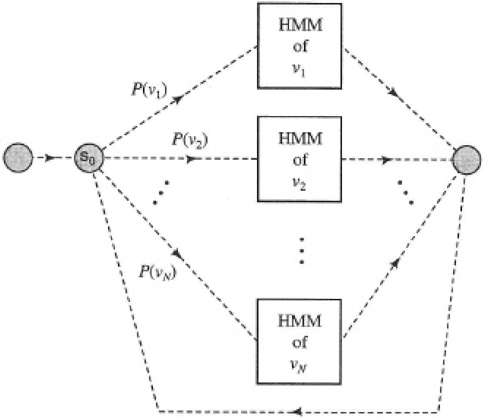
What type of aircraft
What is the fare on flight two seventy two
Show me the flights I've Boston to San Francisco on Monday
What is the cheapest one way
Okay on flight number seven thirty six
What airline leaves earliest
Which airlines from Philadelphia to Dallas
I'd like to leave at nine eight
What airline
How much does it cost
How many stops does Delta flight five eleven o'clock PM that go from
What AM
Is Eastern from Denver before noon
Earliest flight from Dallas
I need to Philadelphia
Describe to Baltimore on Wednesday from Boston
I'd like to depart before five o'clock PM
Which flights do these flights leave after four PM and lunch and <unknown>

The “unknown” word

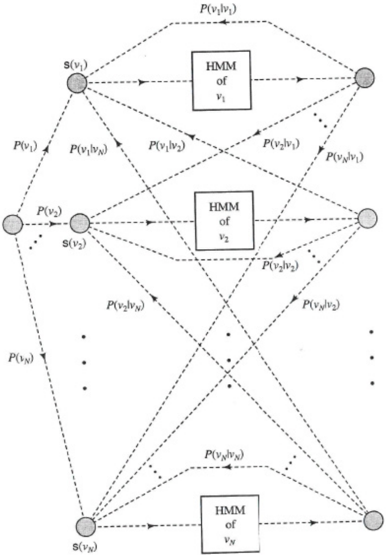
- LMs typically operate with a fixed vocabulary of words
- If we see a new word in test data, we’ll get it wrong
- But we don’t want to get the words around it wrong too...
- The “unknown” word is an extra word often added to the LM vocabulary to give probability to new words

Integration into HMMs: Unigram

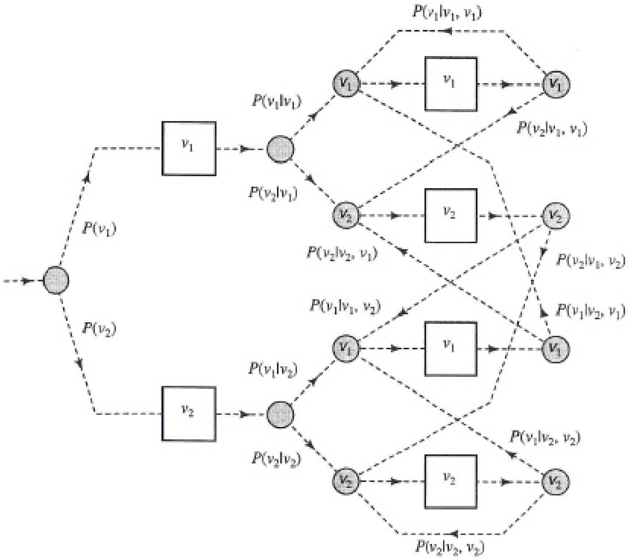
- As with deterministic grammars, we can build a huge HMM representing the entire search space



Integration into HMMs: Bigram



Integration into HMMs: Trigram



Evaluating language models

- One way: Qualitatively (How good do random sentences generated from the LM look?)
- The ultimate way: Task performance (e.g., word error rate)
- Would like some intermediate way...
  - LMs are much quicker to estimate and run than entire speech systems
  - LMs can be estimated independently from text and then used in different tasks
  - Would like a way to test a LM independently of the final task

Evaluating LMs: Cross-entropy

- If X is a discrete random variable taking one of N values with probabilities $p_1, \ldots, p_N$, respectively, then the entropy of X is
$$H(X) = -\sum_{i=1}^{N} p_i \log_2 p_i$$
- The *cross-entropy* of a model $\hat{p}(X)$ with respect to some data set $\mathbf{X} = \{x_1, \ldots, x_n\}$ is $H_{\hat{p}}(\mathbf{X}) = -\frac{1}{n} \log_2 \hat{p}(\mathbf{X})$
- For an *n*-gram LM $\hat{p}(\cdot)$ on a test set $\mathbf{w} = (w_1, \ldots, w_n)$,

$$\begin{aligned}
H_{\hat{p}}(\mathbf{w}) &= -\frac{1}{n} \log_2 \hat{p}(\mathbf{w}) \\
&= -\frac{1}{n} \log_2 \prod_{i=1}^{n} \hat{p}(w_i|\phi(h_i)) \\
&= -\frac{1}{n} \sum_{i=1}^{n} \log_2 \hat{p}(w_i|\phi(h_i))
\end{aligned}$$

Evaluating LMs: Cross-entropy (2)

- Intuition: This is the number of bits per word needed to encode this data set using the model
- For English texts, cross-entropy ranges from around 6 to 10 bits/word

Evaluating LMs: Perplexity

- Perplexity is related to the cross-entropy via $PP_p(\mathbf{w}) = 2^{H_p(\mathbf{w})}$
- For most purposes, lower entopy/perplexity $\Rightarrow$ lower uncertainty about the following word $\Rightarrow$ better language model
- For English text, perplexity ranges from around 25 to several 100s.
- Exercise: What is the perplexity of a uniform LM?
- Answer: the vocabulary size N
- Intuition: Perplexity is the average number of words possible after a given history (the average *branching factor* of the LM)

Evaluating LMs on different domains

Domain	Size	Type	Perplexity
Digits	11	All word	11
Resource Management	1,000	Word-pair	60
		Bigram	20
Air Travel Understanding	2,500	Bigram	29
		4-gram	22
WSJ Dictation	5,000	Bigram	80
		Trigram	45
	20,000	Bigram	190
		Trigram	120
Switchboard Human-Human	23,000	Bigram	109
		Trigram	93
NYT Characters	63	Unigram	20
		Bigram	11
Shannon Letters	27	Human	~ 2