COM4511 Speech Technology: Advanced Acoustic Models

Anton Ragni

February 24, 2020

Statistical Speech Recognition

- Given a parameterised audio sequence, infer the underlying latent representation
  - parameterised audio: sequences of feature vectors $\boldsymbol{O}_{1:T} = \boldsymbol{o}_1, \ldots, \boldsymbol{o}_T$
  - latent representation: sequences of words $\boldsymbol{w}_{1:L} = w_1, \ldots, w_L$
- Options for inference
  - maximum-a-posteriori
$$\boldsymbol{w}^* = \arg\max_{\boldsymbol{w}} \left\{ P(\boldsymbol{w}|\boldsymbol{O}_{1:T}) \right\}$$
    - yields most probable sequence of words (sentence-level)
  - minimum Bayes' risk
$$\boldsymbol{w}^* = \arg\min_{\boldsymbol{w}} \left\{ \sum_{\boldsymbol{w}'} P(\boldsymbol{w}'|\boldsymbol{O}_{1:T}) \mathcal{L}(\boldsymbol{w}, \boldsymbol{w}') \right\}$$
    - yields sequence of words with the smallest expected loss (word or character level)
- Need to know:
  - how to model the posterior and perform inference (search)

Speech recognition centric view – later lectures will cover speech synthesis

Anton Ragni

Posterior Probability Modelling

- Generative approach models posterior indirectly

$$P(\boldsymbol{w}_{1:L}|\boldsymbol{O}_{1:T}) = \frac{1}{p(\boldsymbol{O}_{1:T})} p(\boldsymbol{O}_{1:T}|\boldsymbol{w}_{1:L}) P(\boldsymbol{w}_{1:L})$$

  - standalone acoustic model $p(\boldsymbol{O}_{1:T}|\boldsymbol{w}_{1:L})$

- Discriminative approach models posterior directly
  - acoustic model is integrated into posterior — integrated or end-to-end approaches
  - BUT so is the language model

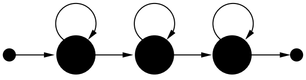
- Both approaches model the posterior probability
  - what is the difference? which one is better?

Anton Ragni

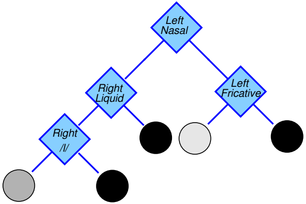
State of The Art in Generative Modelling — All Variants of HMM

$\sim$ 1980	GMM-HMM	L. R. Rabiner, "A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition", Proc IEEE, 1989
$\sim$ 2000	Tandem (FFNN)	H. Hermansky *et al*, "Tandem Connectionist Feature Extraction for Conventional HMM systems", Proc ICASSP, 2000
$\sim$ 2010	Hybrid (FFNN)	G. Hinton *et al*, "Deep Neural Networks for Acoustic Modeling in Speech Recognition", IEEE Sig Proc Mag, 2012
$\sim$ 2013	Hybrid (RNN)	H. Sak *et al*, "Long Short-Term Memory Recurrent Neural Network Architectures for Large Scale Acoustic Modeling", Proc Interspeech, 2014
$\sim$ 2015	Hybrid (CNN and RNN)	T. Sainath *et al*, "Convolutional, Long Short-Term Memory, fully connected Deep Neural Networks", Proc ICASSP, 2015 V. Peddinti *et al*, "Low latency acoustic modeling using temporal convolution and LSTMs", IEEE Sig Proc Let, 2018
$\sim$ 2018	Hybrid (CNN)	D. Povey, "Semi-Orthogonal Low-Rank Matrix Factorization for Deep Neural Networks", Proc Interspeech, 2018

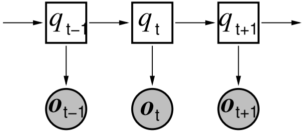
Hidden Markov Models



(a) Standard ASR topology



(b) Phonetic decision trees



(c) Dynamic Bayesian Network

- Observations conditionally independent of other observations given state
- States conditionally independent of other states given past state

$$p(\mathbf{O}_{1:T}|\boldsymbol{w}_{1:L}) = \sum_{\boldsymbol{q}_{1:T} \in \boldsymbol{Q}_{1:T}^{(\boldsymbol{w}_{1:L})}} \prod_{t=1}^{T} p(\boldsymbol{o}_t|q_t) P(q_t|q_{t-1})$$

- transition probabilities, $P(q_t|q_{t-1})$, probability mass functions (discrete distributions)
- output distributions, $p(\boldsymbol{o}_t|q_t)$, probability density functions (continuous distributions)

Anton Ragni

Gaussian Mixture Models

- Form of probability density function

$$p(\boldsymbol{o}|q) = \sum_{m=1}^{M} c_m \mathcal{N}(\boldsymbol{o}; \boldsymbol{\mu}_m, \boldsymbol{\Sigma}_m)$$

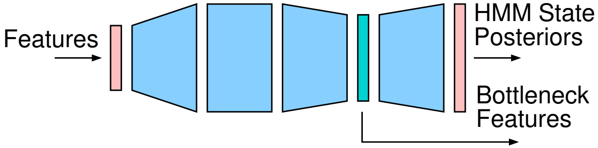
  - mixture "weights" $\boldsymbol{c}_{1:M}$ such that $c_m \geq 0$ for all m and $\sum_{m=1}^{M} c_m = 1$
  - how many parameters are in the model?
- Simple form of Gaussian distribution enables easy adjustment
  - speakers, noise and environment, e.g. maximum likelihood linear regression

$$\boldsymbol{\mu}_m^{(s)} = \boldsymbol{A}^{(s)}\boldsymbol{\mu}_m + \boldsymbol{b}^{(s)}, \quad \boldsymbol{\Sigma}^{(s)} = \boldsymbol{H}^{(s)^{\mathsf{T}}}\boldsymbol{\Sigma}_m\boldsymbol{H}^{(s)}$$

  - decision boundary, e.g., maximum mutual information

$$\mathcal{L}(\mathcal{D}; \boldsymbol{\theta}) = -\frac{1}{|\mathcal{D}|}\sum_{(\boldsymbol{w},\boldsymbol{O})\in\mathcal{D}} \log(P(\boldsymbol{w}|\boldsymbol{O}))$$

Tandem



- Feed-forward neural network

$$\boldsymbol{h}^{(l)} = \phi^{(l)}(\boldsymbol{A}^{(l)}\boldsymbol{h}^{(l-1)} + \boldsymbol{b}^{(l)})$$

  - initialise $\boldsymbol{h}^{(0)} = \boldsymbol{o}$, terminate $P(q = i|\boldsymbol{o}) = h_i^{(L)}$, $\phi^{(L)}$ softmax, $\phi^{(l<L)}$ sigmoid/ReLU

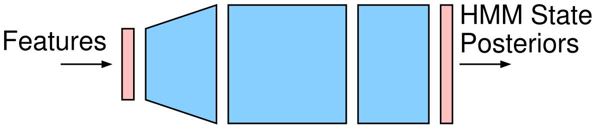
- Augment hand-crafted features with features learnt by the neural network
  - options available what data to use (matched, mismatched, multi-lingual)

- BUT learnt features may be correlated
  - if Gaussians with diagonal covariance matrices used need to decorrelate

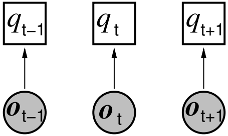
$$\hat{\boldsymbol{o}} = \begin{bmatrix} \boldsymbol{I} & \boldsymbol{I} \\ \boldsymbol{I} & \boldsymbol{S} \end{bmatrix} \begin{bmatrix} \boldsymbol{o} \\ \boldsymbol{h}^{(l)} \end{bmatrix}$$

Anton Ragni

Hybrid



(a) Standard ASR architecture

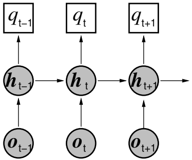


(b) DNN DBN

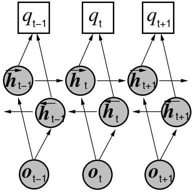
- Use feed-forward neural network to predict HMM state posteriors
  - efficient use of parameters (c.f. decision trees)
  - enables complex forms of classifiers (c.f. lecture on neural networks)
  - less sensitive to correlated features (c.f. MFCC)
- Possible to integrate into generative and discriminative sequence models
  - generative: HMM (discussed later)
  - discriminative: CTC (discussed later)
- Other types of neural networks are currently more popular
  - recurrent neural network, convolutional neural network and their combinations

Anton Ragni

Recurrent Neural Networks



(a) (Unidirectional) RNN



(b) Bidirectional RNN

- Introduce dependence on all past observations using learnt history representation

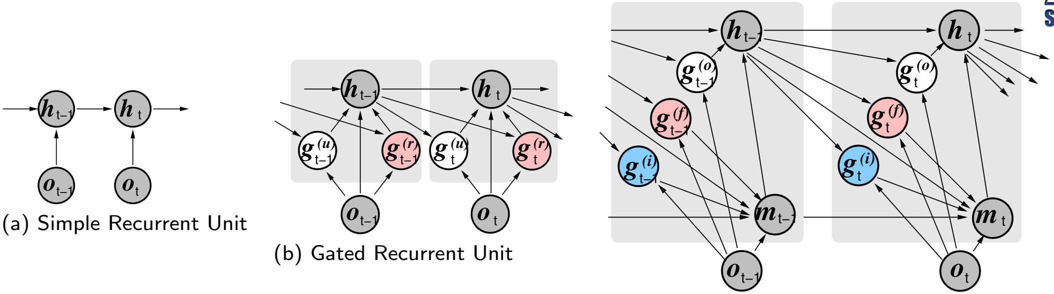
$$P(\boldsymbol{q}_{1:T}|\boldsymbol{O}_{1:T}) = \prod_{t=1}^{T} P(q_t|\boldsymbol{q}_{1:t-1}, \boldsymbol{O}_{1:T}) \approx \prod_{t=1}^{T} P(q_t|\boldsymbol{O}_{1:t}) \approx \prod_{t=1}^{T} P(q_t|\boldsymbol{h}_t) \qquad (1)$$

  - assumes future observations are of no use
- Use another recurrent cell to condition on future observations (bi-directional RNN)

$$P(\boldsymbol{q}_{1:T}|\boldsymbol{O}_{1:T}) \approx \prod_{t=1}^{T} P(q_t|\overrightarrow{\boldsymbol{h}}_t, \overleftarrow{\boldsymbol{h}}_t)$$

  - BUT states remain conditionally independent

Recurrent Units



(a) Simple Recurrent Unit

(b) Gated Recurrent Unit

(c) Long Short-Term Memory

- Simple recurrent unit is known to have unstable behaviour
  - use "gates" to pass or block

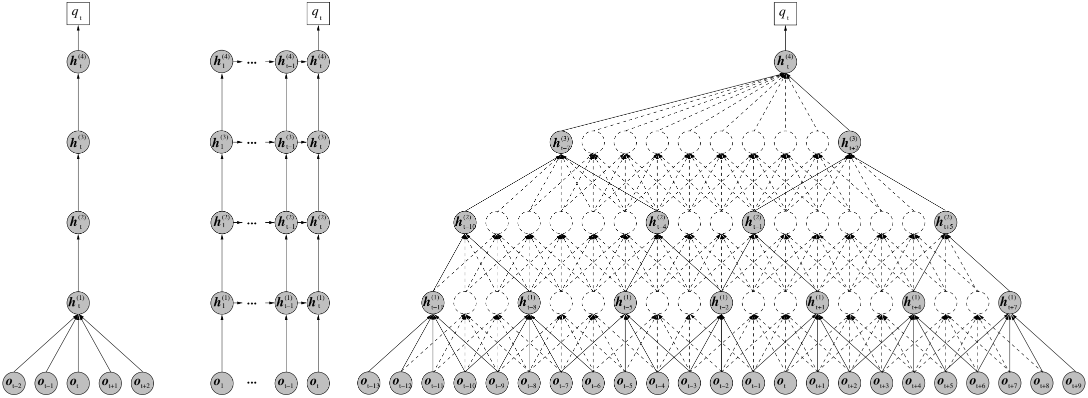
$$\boldsymbol{g}_t = \boldsymbol{\sigma}(\boldsymbol{A}^{(g)}\boldsymbol{o}_t + \boldsymbol{C}^{(g)}\boldsymbol{h}_{t-1} + \boldsymbol{b}^{(g)}), \quad \text{where } \sigma \text{ is a sigmoid}$$

- Multiple "smart" recurrent units can be devised, e.g. GRU

$$\boldsymbol{h}_t = \boldsymbol{g}_t^{(u)} \odot \boldsymbol{h}_{t-1} + (\boldsymbol{1} - \boldsymbol{g}_t^{(u)}) \odot \phi(\boldsymbol{A}\boldsymbol{o}_t + \boldsymbol{C}(\boldsymbol{g}_t^{(r)} \odot \boldsymbol{h}_{t-1}) + \boldsymbol{b})$$

Anton Ragni

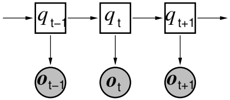
Time-Delay Neural Networks Models



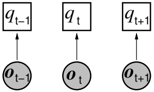
(a) Feed-Forward (b) Recurrent (c) Time Delay

- Long-term dependency modelling using standard networks is challenging
  - feed-forward models are highly inefficient
  - recurrent models are hard to train (full history, instability)
- Use a sub-sampled pyramidal structure to increase scope of possible dependencies
  - sharing parameteres in each layer is akin to a filter (convolutional network!)

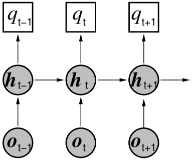
Integration with HMMs



(a) HMM DBN



(b) DNN DBN



(c) RNN Pseudo-DBN

- Feed-forward types inherit standard conditional independence assumptions

$$p(\boldsymbol{O}_{1:T}|\boldsymbol{q}_{1:T}) = \prod_{t=1}^{T} p(\boldsymbol{o}_t|\boldsymbol{O}_{1:t-1}, \boldsymbol{q}_{1:T}) \approx \prod_{t=1}^{T} p(\boldsymbol{o}_t|q_t) = \prod_{t=1}^{T} \frac{\overbrace{P(q_t|\boldsymbol{o}_t)}^{FFNN}\, p(\boldsymbol{o}_t)}{P(q_t)}$$

- Recurrent types relax conditional independence assumptions of observations

$$p(\boldsymbol{O}_{1:T}|\boldsymbol{q}_{1:T}) = \prod_{t=1}^{T} p(\boldsymbol{o}_t|\boldsymbol{O}_{1:t-1}, \boldsymbol{q}_{1:T}) \approx \prod_{t=1}^{T} p(\boldsymbol{o}_t|\boldsymbol{O}_{1:t-1}, q_t) = \prod_{t=1}^{T} \frac{\overbrace{P(q_t|\boldsymbol{O}_{1:t})}^{RNN}\, p(\boldsymbol{o}_t|\boldsymbol{O}_{1:t-1})}{P(q_t|\boldsymbol{O}_{1:t-1})}$$

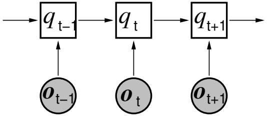
  - BUT not the assumptions about states

Anton Ragni

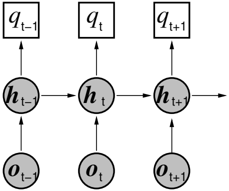
State of The Art in Discriminative Modelling

$\sim$ 2010	SCRF/CAug	G. Zweig *et al*, "A Segmental CRF Approach to Large Vocabulary Continuous Speech Recognition", ASRU, 2009 A. Ragni *et al*, "Derivative kernels for noise-robust ASR", ASRU, 2011
$\sim$ 2012	CTC	A. Graves *et al*, "Towards End-to-End Speech Recognition with Recurrent Neural Networks", ICML, 2014
$\sim$ 2014	Encoder-Decoder	I. Sutskever, "Sequence to Sequence Learning with Neural Networks", NIPS, 2014
$\sim$ 2016	Attention	W. Chan, "Listen, Attend and Spell", ICASSP, 2016
$\sim$ 2018	RNN-Transducer	Y. He *et al*, "Streaming end-to-end speech recognition for mobile devices", ICASSP, 2019
$\sim$ 2018	Transformer	A. Zayer *et al*, "A Comparison of Transformer and LSTM Encoder Decoder Models for ASR", ASRU, 2019

Connectionist Temporal Classification (CTC)



(a) MEMM DBN



(b) CTC Pseudo-DBN

- Direct approach to using neural network for character sequence prediction

$$P(\boldsymbol{w}_{1:L}|\boldsymbol{O}_{1:T}) \approx \sum_{\boldsymbol{q}_{1:T}\in\boldsymbol{Q}^{(\boldsymbol{w}_{1:L})}} P(\boldsymbol{q}_{1:T}|\boldsymbol{O}_{1:T}) \approx \sum_{\boldsymbol{q}_{1:T}\in\boldsymbol{Q}^{(\boldsymbol{w}_{1:L})}} \prod_{t=1}^{T} P(q_t|\boldsymbol{h}_t)$$

  - relaxes assumptions about observations but not about latent variables
  - possible to use MEMM (HMM) style forward-backward algorithm

- Need to decide on the form of latent variable model (LVM)
  - only need to be able to generate latent variable sequences — options available?

Example: MEMM/HMM Latent Variable Model

- Latent variable model "rules"
  1. each symbol must appear at least once
  2. symbol order must be preserved
  3. must have mechanism to track symbol changes
- Given character sequence w_1, w_2, w_2, w_3, possible latent variable sequences are

Length (T)	Latent Variable Sequences $\boldsymbol{q}_{1:T}$
4	$w_1, w_2, w_2, w_3; \ldots$
5	$w_1, w_2, w_2, w_2, w_3; \ldots$
6	$w_1, w_2, w_2, w_2, w_2, w_3; \ldots$
7	$w_1, w_2, w_2, w_2, w_2, w_2, w_3; \ldots$
8	$w_1, w_2, w_2, w_2, w_2, w_2, w_2, w_3; \ldots$
9	$w_1, w_1, w_2, w_2, w_2, w_2, w_2, w_2, w_3; \ldots$
$\vdots$	$\ldots$

  - explicit start/end states enable to know symbol boundaries
  - write a regular expression to express all possible latent variable sequences

Anton Ragni

Example: CTC "Latent Variable Model"

- Two observations can be made about latent variable handling in MEMM/HMM
  - symbol change is ambiguous only for two identical symbols
  - forcing each q_t to take one of $\boldsymbol{w}_{1:L}$ values may be problematic
- Introduce a new symbol ϵ to play the role of a delimiter and an uncertain symbol
  - encourage each q_t to take values in $\boldsymbol{w}_{1:L}$ only if highly certain
  - BUT has to artificially enforce ϵ between two identical symbols
- Given character sequence w_1, w_2, w_2, w_3, possible latent variable sequences are

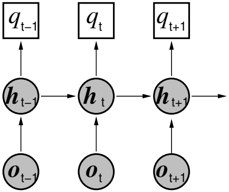
Length (T)	Latent Variable Sequences $\boldsymbol{q}_{1:T}$
4	—
5	$w_1, w_2, \epsilon, w_2, w_3; \ldots$
6	$w_1, \epsilon, w_2, \epsilon, w_2, w_3; \ldots$
7	$w_1, \epsilon, \epsilon, w_2, \epsilon, w_2, w_3; \ldots$
8	$w_1, w_1, \epsilon, \epsilon, w_2, \epsilon, w_2, w_3; \ldots$
9	$w_1, w_1, \epsilon, \epsilon, w_2, \epsilon, w_2, w_3, w_3; \ldots$

  - no need to have an explicit latent variable model
  - write a regular expression to express all possible latent variable sequences

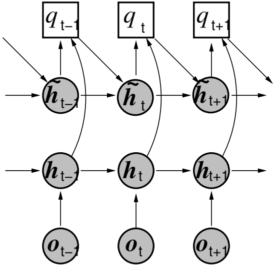
SPandH

Anton Ragni

Recurrent Neural Network Transducer (RNN-T)



(a) CTC



(b) RNN-T

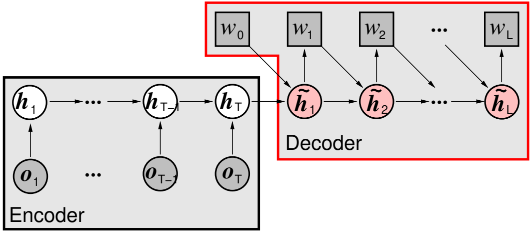
- Break conditional independence assumption among latent variables

$$P(\boldsymbol{w}_{1:L}|\boldsymbol{O}_{1:T}) \approx \sum_{\boldsymbol{q}_{1:T} \in \boldsymbol{Q}^{(\boldsymbol{w}_{1:L})}} \prod_{t=1}^{T} P(q_t|\boldsymbol{q}_{1:t-1}, \boldsymbol{O}_{1:t}) \approx \sum_{\boldsymbol{q}_{1:T} \in \boldsymbol{Q}^{(\boldsymbol{w}_{1:L})}} \prod_{t=1}^{T} P(q_t|\tilde{\boldsymbol{h}}_t, \boldsymbol{h}_t)$$

  - reported to yield competitive performance given large amounts of data

Anton Ragni

Encoder-Decoder



Predict next word using recursion

$$\tilde{\boldsymbol{h}}_l = \phi(\tilde{\boldsymbol{h}}_{l-1}, w_{l-1})$$

Options for setting $\tilde{\boldsymbol{h}}_0$:

- use output of another recursion

$$\boldsymbol{h}_T = \phi(\boldsymbol{h}_{T-1}, \boldsymbol{o}_T)$$

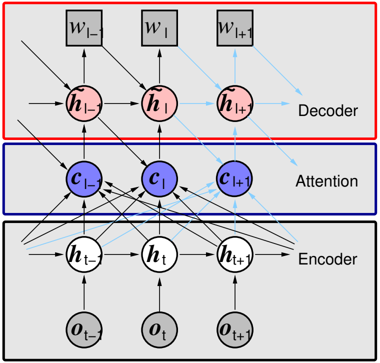
- name other common options

- Use recurrent units to handle variable length observation and word sequences

$$P(\boldsymbol{w}_{1:L}|\boldsymbol{O}_{1:T}) = \prod_{l=1}^{L} P(w_l|\boldsymbol{w}_{1:l-1}, \boldsymbol{O}_{1:T}) \approx \prod_{l=1}^{L} P(w_l|\boldsymbol{w}_{1:l-1}, \boldsymbol{h}_T) \approx \prod_{l=1}^{L} P(w_l|\tilde{\boldsymbol{h}}_l)$$

  - encoder: maps observation sequence $\boldsymbol{O}_{1:T}$ to fixed length representation $\boldsymbol{h}_T$
  - decoder: generates word sequence $\boldsymbol{w}_{1:L}$ given $\boldsymbol{h}_T$
- BUT hard to ensure relevant information propagated/used for prediction

Attention



Position-dependent history

$$\boldsymbol{c}_l = \sum_{t=1}^{T} \alpha_{l,t} \boldsymbol{h}_t, \quad \text{where} \quad \alpha_{l,t} = \frac{\exp(z_{l,t})}{\sum_{t=1}^{T} \exp(z_{l,t})}$$

Unnormalised attention weights

$$z_{l,t} = \boldsymbol{d}^{\mathsf{T}} \phi(\boldsymbol{A}\tilde{\boldsymbol{h}}_{l-1} + \boldsymbol{C}\boldsymbol{h}_t)$$

- "IR": query $\tilde{\boldsymbol{h}}_{l-1}$, key $\boldsymbol{h}_t$, values $\boldsymbol{h}_{1:T}$

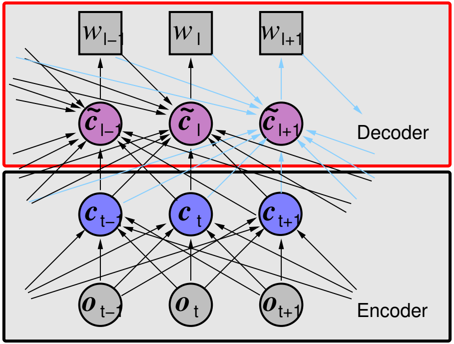
(other forms possible)

- Improve encoder-decoder by position-dependent input representation

$$P(\boldsymbol{w}_{1:L}|\boldsymbol{O}_{1:T}) = \prod_{l=1}^{L} P(w_l|\boldsymbol{w}_{1:l-1}, \boldsymbol{O}_{1:T}) \approx \prod_{l=1}^{L} P(w_l|\boldsymbol{w}_{1:l-1}, \boldsymbol{c}_l) \approx \prod_{l=1}^{L} P(w_l|\tilde{\boldsymbol{h}}_l)$$

  - reported to yield competitive performance given large amounts of data

Transformer (Not a movie!)



- Replace all recurrent units with attention
  - multi-head attention: multiple parallel mechanisms to increase modelling power
  - positional encoding: add positional offset to encode temporal information

$$P(\boldsymbol{w}_{1:L}|\boldsymbol{O}_{1:T}) = \prod_{l=1}^{L} P(w_l|\boldsymbol{w}_{1:l-1}, \boldsymbol{O}_{1:T}) \approx \prod_{l=1}^{L} P(w_l|\boldsymbol{w}_{1:l-1}, \boldsymbol{c}_{1:T}) \approx \prod_{l=1}^{L} P(w_l|\tilde{\boldsymbol{c}}_l)$$

- Reported to yield competitive performance given large amounts of data

Anton Ragni

Integration with External Language Models

- Generative approaches provide a natural framework to integrate language models

$$P(\boldsymbol{w}_{1:L}|\boldsymbol{O}_{1:T}) \propto p(\boldsymbol{O}_{1:T}|\boldsymbol{w}_{1:L})P(\boldsymbol{w}_{1:L})$$

  - enables to plug-in any suitable language model

- Less obvious how to integrate language model into discriminative models
  - heuristic probability combination

$$s(\boldsymbol{w}_{1:L}, \boldsymbol{O}_{1:T}) = P(\boldsymbol{w}_{1:L}|\boldsymbol{O}_{1:T})P(\boldsymbol{w}_{1:L})$$

  - fusion between decoder and language model recurrent units

$$\tilde{\boldsymbol{h}}_l = \phi(\tilde{\boldsymbol{h}}_{l-1}, \tilde{\boldsymbol{h}}_{l-1}^{(LM)}, w_{l-1})$$

  - multi-task training, pre-training and many more!

Summary

- The past two lectures have
  - introduced hidden Markov models (HMM)
  - discussed how HMMs can be used as an acoustic model
- This lecture has discussed more complex forms of acoustic models
  - generative models (Tandem, Hybrids)
  - discriminative models (CTC, encoder-decoder, transformer)
- Next lectures will look at language models