CS 533: Natural Language Processing

Copy Mechanism, Relation-Aware Self-Attention, Hidden Markov Models

Karl Stratos

Rutgers University

Review: Conditional Language Models

- Language model (LM) conditioning on $x = (x_1 \ldots x_T)$

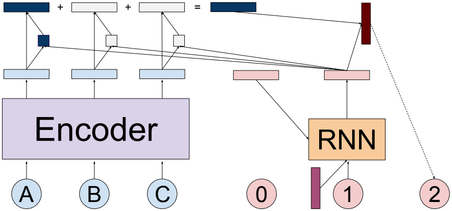
$$p_\theta(y_1 \ldots y_{T'}|x) = \prod_{t'=1}^{T'+1} p_\theta(y_{t'}|x, y_{<t'})$$

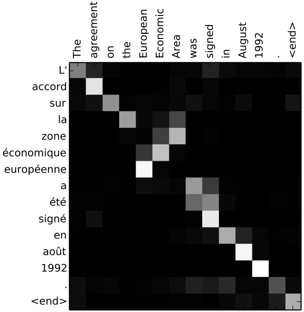
- Learnable modules
  - **Encoder.** $\textbf{enc}_\theta : \mathcal{V}^T \rightarrow \mathbb{R}^{T \times d}$ contextualizes source token embeddings of x (e.g., BiLSTM, Transformer encoder)
  - **Decoder.** $\textbf{dec}_\theta : \mathbb{R}^{T \times d} \times \mathcal{V}^{t'-1} \rightarrow \mathbb{R}^V$ computes logits for next word given source encodings and target history via attention to source encodings (e.g., recurrent, Transformer decoder)
- Encoder-decoder/sequence-to-sequence (seq2seq): Train encoder & decoder jointly to optimize a function of

$$p_\theta(y_{t'}|x, y_{<t'}) = \mathrm{softmax}_{y_{t'}}(\textbf{dec}_\theta(\textbf{enc}_\theta(x), y_{<t'}))$$

Review: Stepwise Cross-Attention

- Example: RNN decoder with input feeding





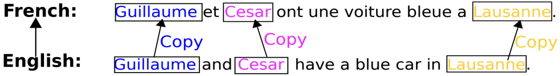
- Learns to attend to right source positions, without supervision. Visualization for translating English to French (Bahdanau et al., 2016)
- Transformer decoder (Vaswani et al., 2017): No recurrent or convolutional layers, entirely based on attention with a position-shared feedforward

The Unknown Word Problem

- Target text may contain rare words like
  - Proper names: Lausanne, Cesar, Guilaume, ...
  - Numbers/values: 103, 95, 42, 3.141592, 3.141593, ...
- Decoder needs these in target vocab $\mathcal{V}$ to generate at all!
  - Note target vocab may be distinct from source vocab $\mathcal{V}_{\mathrm{src}}$ in general (e.g., translation)
- Brute-force: Include all word types in $\mathcal{V}$? Not practical
  - By Zipf's Law, most words will have extremely low probabilities
  - Never enough: Guilaumé? 3.141594? Not seen in training data
- Simple/naive approach: Threshold vocab by frequency
  - Keep top-k (e.g., $k = 100000$) most frequent types in $\mathcal{V}$ and replace all other types ("OOV") with special token $\langle \mathrm{unk} \rangle$ in training
  - Problem: Model predicts $\langle \mathrm{unk} \rangle$ at test time (e.g., "$\langle \mathrm{unk} \rangle$ and $\langle \mathrm{unk} \rangle$ have a blue car in $\langle \mathrm{unk} \rangle$").
  - Can be postprocessed, but can we do better?

Copy Mechanism

- Idea: Unknown target words likely to be **copied** from source sentence somewhere
- Example: translation (Gulcehre et al., 2016)



- Example: data-to-text generation (Wiseman et al., 2017)

TEAM	WIN	LOSS	PTS
Heat	11	12	103
Hawk	7	15	95

The Atlanta Hawks defeated the Miami Heat, 103-95, at Philips Arena on Wednesday...

- Approaches: Data pre-processing, attention-based
- Non-copy approaches
  - Subword tokenization (e.g., BPE): No "unknown" words, but sequences longer and may also benefit from copy mechanism
  - Scaling softmax to accommodate bigger $\mathcal{V}$ (e.g., hierarchical softmax, sampling-based methods)

Data Pre-Processing Approach (Luong et al., 2015)

- Original data: Apply an unsupervised aligner to get alignments
  - The ecotax portico in Pont-de-Buis
  - Le portique écotaxe de Pont-de-Buis
- Conventional pre-processing
  - The $\langle \text{unk} \rangle$ portico in $\langle \text{unk} \rangle$
  - Le $\langle \text{unk} \rangle$ $\langle \text{unk} \rangle$ de $\langle \text{unk} \rangle$
- Copyable Model pre-processing
  - The $\langle \text{unk} \rangle_1$ portico in $\langle \text{unk} \rangle_2$
  - Le $\langle \text{unk} \rangle_0$ $\langle \text{unk} \rangle_1$ de $\langle \text{unk} \rangle_2$
- Positional All Model pre-processing
  - The $\langle \text{unk} \rangle$ portico in $\langle \text{unk} \rangle$
  - Le p_0 $\langle \text{unk} \rangle$ p_{-1} $\langle \text{unk} \rangle$ p_1 de p_0 $\langle \text{unk} \rangle$ p_{-1}
- Positional Unknown Model pre-processing
  - The $\langle \text{unk} \rangle$ portico in $\langle \text{unk} \rangle$
  - Le $\langle \text{unk} \rangle_1$ $\langle \text{unk} \rangle_{-1}$ de $\langle \text{unk} \rangle_1$

Attention-Based Approaches

- Data pre-processing approach: Simple and effective (1-2 points improvement over strong NMT baselines)
- Limitations
  - Requires an external word aligner in the pipeline
  - Fixed-size window ($\langle \mathrm{unk} \rangle_{-7} \ldots \langle \mathrm{unk} \rangle_{7}$), can't handle copy from far away in source sequence
- Idea: Make the model learn *when* and *what* to copy without supervision, by attention
- Pointer networks (Vinyals et al., 2015): Only what to copy
- CopyNet (Gu et al., 2016): Both when and what to copy, applied on summarization
- Concurrent work by Gulcehre et al., 2016: Different modeling details, applied on both translation and summarization
- When to copy: Modeled by a "switching network" (learned jointly)

Conditional LM with a Copy Mechanism

- Single training example now consists of

$$x = (x_1 \dots x_T) \qquad y = (y_1 \dots y_{T'}) \qquad z = (z_1 \dots z_{T'})$$

  where $z_{t'} \in \{0, 1\}$ is 1 iff $y_{t'}$ is copied from x
- Assume for now that z is observed
  - Just decide to set $z_{t'} = 1$ if $y_{t'}$ appears in x somewhere.
- Conditional LM with a copy mechanism

$$p_\theta(y, z|x) = \prod_{t'=1}^{T'+1} p_\theta(y_{t'}, z_{t'}|x, y_{<t'}, z_{<t'})$$

- Further decomposition by the chain rule

$$p_\theta(y_t, z_{t'}|x, y_{<t'}, z_{<t'}) = \underbrace{p_\theta(z_{t'}|x, y_{<t'}, z_{<t'})}_{\text{"switching network"}} \times p_\theta(y_{t'}|x, y_{<t'}, z_{\leq t'})$$

Parameterization

- Switching network

$$p_\theta(1|x, y_{<t'}, z_{<t'}) = \sigma\left(f_\theta(x, y_{<t'}, z_{<t'})\right)$$
$$p_\theta(0|x, y_{<t'}, z_{<t'}) = 1 - \sigma\left(f_\theta(x, y_{<t'}, z_{<t'})\right)$$

$f_\theta(x, y_{<t'}, z_{<t'}) \in \mathbb{R}$ computed from current state (e.g., h_t if RNN, current embedding if Transformer)

- If $z_{t'} = 1$, "dynamic LM" with vocab $\{w \in x\}$

$$p_\theta(y_{t'} = w|x, y_{<t'}, z_{\leq t'}) = \sum_{t=1:\ x_t = w}^{T} \underbrace{A^\theta_{t,t'}}_{\text{attention from } t'\text{-th target to } t\text{-th source}}$$

- If $z_{t'} = 0$, vocab $\mathcal{V}$

$$p_\theta(y_{t'} = w|x, y_{<t'}, z_{\leq t'}) = \underbrace{p_\theta(y_{t'} = w|x, y_{<t'})}_{\text{usual next word probability}}$$

Supervised vs Unsupervised Loss

- Supervised training: Maximize $\log p_\theta(y, z|x)$ in training data
  - Inference: At each step t', consider all

$$p_\theta(w, 1|x, y_{<t'}, z_{<t'}) \qquad \forall w \in \mathcal{V}$$
$$p_\theta(w, 0|x, y_{<t'}, z_{<t'}) \qquad \forall w \in x$$

- Unsupervised training: Maximize $\log p_\theta(y|x)$ in training data

$$p_\theta(y_{t'}|x, y_{<t'}) = \sum_{z \in \{0,1\}} p_\theta(y_{t'}, z|x, y_{<t'})$$
$$= \sigma\left(f_\theta(x, y_{<t'}, z_{<t'})\right)\left(\sum_{t=1:\ x_t = y_{t'}}^{T} A^\theta_{t,t'}\right) +$$
$$(1 - \sigma\left(f_\theta(x, y_{<t'}, z_{<t'})\right))p_\theta(y_{t'} = w|x, y_{<t'})$$

Switching network f_θ trained without supervision, inference remains the same

Illustration

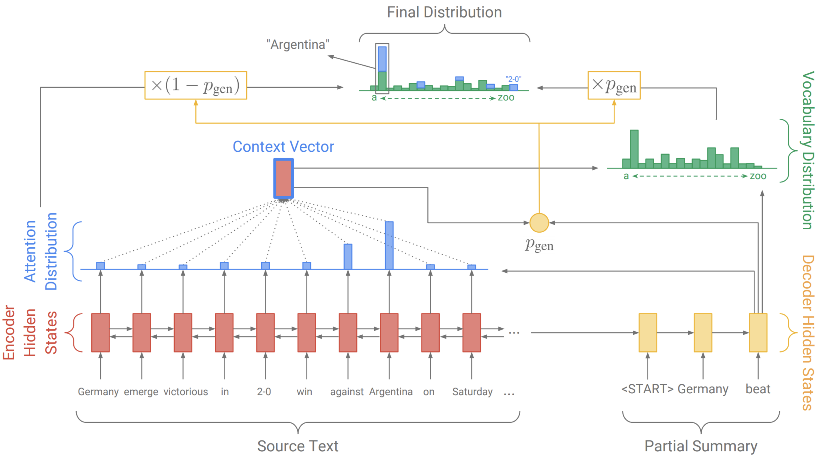
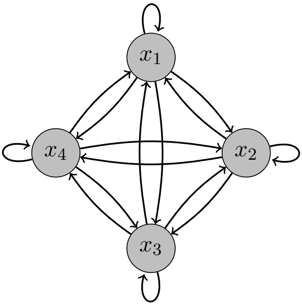
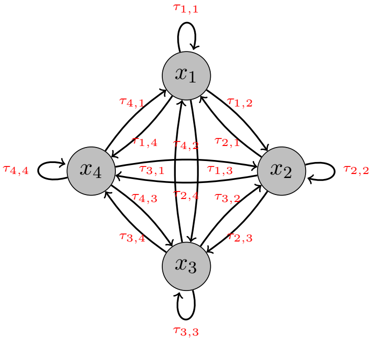


Image credit: See et al. (2017)

Self-Attention as a Fully Connected Directed Graph

- Self-attention viewed as a fully connected directed graph
- Natural generalization: Incorporate edge types in the model





- Example edge types: Relative positions, relation between table cells (e.g., cell-column, cell-row)

Relation-Aware Self-Attention (Shaw et al., 2018)

- Extra parameters in the multi-head attention module
  - $b^K_\tau \in \mathbb{R}^{d/H}$ for every relation type τ
  - $b^V_\tau \in \mathbb{R}^{d/H}$ for every relation type τ
- Self-attention weight from $x_{t'}$ to x_t with relation $\tau_{t',t}$ under head h

$$l^h_{t',t} = \frac{q^h_{t'} \cdot (k^h_t + b^K_{\tau_{t',t}})}{d/H}$$

  Probabilities: $(\alpha^h_{t',1} \ldots \alpha^h_{t',T}) = \mathrm{softmax}(l^h_{t',1} \ldots l^h_{t',T})$

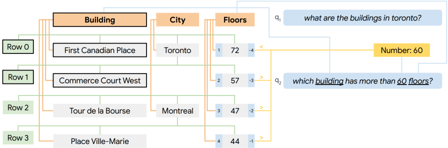
- Answer value

$$a^h_{t'} = \sum_{t=1}^{T} \alpha^h_{t,t'} \left(v^h_t + b^V_{\tau_{t',t}} \right)$$

- Relation bias is shared across all heads. Efficient batch computation still possible by construction

Applications of Relation-Aware Self-Attention

- Relative position encoding (Shaw et al., 2018)
  - Original Transformer: Add constant (or learnable) absolute position embeddings at input vectors
  - Now: For some k (e.g., $k = 8$), use $2k + 1$ relation types representing local distances
  - Tokens beyond window clipped to k or $-k$
  - Can entirely replace additive position embeddings, even modest improvement
  - Value bias b_τ^V found unnecessary given key bias b_τ^T (for MT)
- Relation between tokens in structured input (Müller et al., 2019)
  - Task: question answering from a table (represented as a flat sequence of words)
  - Idea: Distinguish relations between table cells, row header, column header, question, etc.



Sequence Labeling/Tagging

- Switching gears, we'll consider the sequence labeling (aka. tagging) problem.
- Task: Given sentence $x_1 \ldots x_T \in \mathcal{V}$, output a correct label sequence $y_1 \ldots y_T \in \mathcal{Y}$
- Many applications: part-of-speech tagging, named-entity recognition
- This is a **structured prediction** problem: Output space is $\mathcal{Y}^T$ possible label sequences
- Why not just frame it as seq2seq?
  - Seq2seq needs a lot of data, and is typically very challenging to train well (lots of engineering efforts)
  - In contrast, we can exploit **conditional independence assumptions** to derive exact and effective algorithms
  - In tagging, exact inference called "Viterbi", exact marginalization called "forward". Both dynamic programming

Example: Part-Of-Speech (POS) Tagging

- **Task.** Given a sentence, output a sequence of POS tags.
- **Ambiguity.** A word can have many possible POS tags.

  `the/DT man/NN saw/VBD the/DT cut/NN`

  `the/DT saw/NN cut/VBD the/DT man/NN`
- **Evaluation.** Per-position accuracy (can consider others, like sentence-level accuracy)
- Definition of POS tags in Penn Treebank (English)

Tag	Description	Example	Tag	Description	Example	Tag	Description	Example
CC	coordinating conjunction	*and, but, or*	PDT	predeterminer	*all, both*	VBP	verb non-3sg present	*eat*
CD	cardinal number	*one, two*	POS	possessive ending	*'s*	VBZ	verb 3sg pres	*eats*
DT	determiner	*a, the*	PRP	personal pronoun	*I, you, he*	WDT	wh-determ.	*which, that*
EX	existential 'there'	*there*	PRP$	possess. pronoun	*your, one's*	WP	wh-pronoun	*what, who*
FW	foreign word	*mea culpa*	RB	adverb	*quickly*	WP$	wh-possess.	*whose*
IN	preposition/ subordin-conj	*of, in, by*	RBR	comparative adverb	*faster*	WRB	wh-adverb	*how, where*
JJ	adjective	*yellow*	RBS	superlatv. adverb	*fastest*	$	dollar sign	*$*
JJR	comparative adj	*bigger*	RP	particle	*up, off*	#	pound sign	*#*
JJS	superlative adj	*wildest*	SYM	symbol	*+,%, &*	“	left quote	‘ or “
LS	list item marker	*1, 2, One*	TO	“to”	*to*	”	right quote	’ or ”
MD	modal	*can, should*	UH	interjection	*ah, oops*	(	left paren	[, (, {, <
NN	sing or mass noun	*llama*	VB	verb base form	*eat*	)	right paren	],), }, >
NNS	noun, plural	*llamas*	VBD	verb past tense	*ate*	,	comma	,
NNP	proper noun, sing.	*IBM*	VBG	verb gerund	*eating*	.	sent-end punc	. ! ?
NNPS	proper noun, plu.	*Carolinas*	VBN	verb past part.	*eaten*	:	sent-mid punc	: ; ... – -

Figure 8.1 Penn Treebank part-of-speech tags (including punctuation).

[45 tags]

(Marcus et al., 1993)

Other definitions: universal tagset (12 tags, language agnostic)

Example: Named-Entity Recognition (NER)

- **Task.** Given a sentence, identify and label all spans that are "named entities"

$$\texttt{...}\ \underbrace{\texttt{John Smith}}_{\text{PER}}\ \texttt{works at}\ \underbrace{\texttt{New York Times}}_{\text{ORG}}\ \texttt{...}$$

- **Reduction to tagging.** "Linearize" labeled spans into a label sequence using "BIO" scheme

  `John/B-PER Smith/I-PER works/O at/O New/B-ORG York/I-ORG Times/I-ORG`

  Number of tagging labels: $2 \times$ number of entity types $+ 1$

West	B-MISC
Indian	I-MISC
all-rounder	O
Phil	B-PER
Simmons	I-PER
took	O
four	O
for	O
38	O
on	O
Friday	O
as	O
Leicestershire	B-ORG
beat	O
Somerset	B-ORG
by	O

CoNLL 2003 dataset, 4 entity types (PER, ORG, LOC, MISC)

NER Evaluation

- Most words are tagged as O (not an entity), so accuracy is meaningless (vacuously high by predicting O always)
- Better metric: precision/recall/F1
- Per-entity F1 score (harmonic mean of precision and recall)

$$F_1(e) = \frac{2p(e)r(e)}{p(e) + r(e)}$$

$$p(e) = \frac{tp(e)}{tp(e) + fp(e)} \times 100 \qquad r(e) = \frac{tp(e)}{tp(e) + fn(e)} \times 100$$

- Global F1 score: Single performance number

$$F_1 = \frac{2pr}{p + r}$$

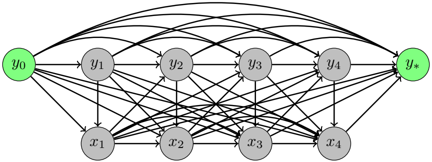
$$p = \frac{tp}{tp + fp} \times 100 \qquad r = \frac{tp}{tp + fn} \times 100$$

Generative Probabilistic Tagger

- Model defines a *joint* distribution $p_\theta(x_1 \ldots x_T, y_1 \ldots y_T)$ over any pairs of sentence and a label sequence.
  - Can generate $x_1 \ldots x_T$, although we will not use the tagger for generation
- By the chain rule

$$p_\theta(x_1 \ldots x_T,\ y_1 \ldots y_T)$$
$$= p_\theta(y_1|y_0) \times p_\theta(x_1|y_0\ y_1) \times p_\theta(y_2|x_1, y_0\ y_1) \times p_\theta(x_2|x_1, y_0\ y_1\ y_2)$$
$$\cdots \times p_\theta(y_T|x_{<T}, y_{<T}) \times p_\theta(x_T|x_{<T}, y_{\leq T}) \times p_\theta(y_*|x_{\leq T}, y_{\leq T})$$

Thus only need to model **transition probabilities** $p_\theta(y_t|x_{<t}, y_{<t})$ and **emission probabilities** $p_\theta(x_t|x_{<t}, y_{\leq t})$



Marginalization and Inference

- Two central calculations in structured prediction
- **Marginalization.** What is the *marginal* probability of $x_1 \ldots x_T$ under the model?

$$\sum_{y_1 \ldots y_T \in \mathcal{Y}^T} p_\theta(x_1 \ldots x_T,\ y_1 \ldots y_T)$$

- **Inference.** Given $x_1 \ldots x_T$, what is the most probable $y_1 \ldots y_T \in \mathcal{Y}^T$ under the model?

$$\begin{aligned} &\operatorname*{arg\,max}_{y_1 \ldots y_T \in \mathcal{Y}^T} p_\theta(y_1 \ldots y_T \mid x_1 \ldots x_T) \\ &= \operatorname*{arg\,max}_{y_1 \ldots y_T \in \mathcal{Y}^T} p_\theta(x_1 \ldots x_T,\ y_1 \ldots y_T) \end{aligned}$$

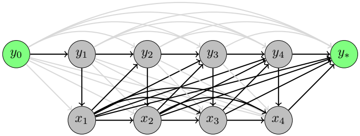
- Generally intractable, that is we must exhaustively enumerate $|\mathcal{Y}|^T$ tag sequences (exponential in length).

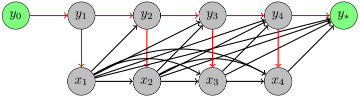
(First-Order) Markov Assumption

- We *define* the model as

$$p_\theta(y_t|x_{<t}, y_{<t}) = p_\theta(y_t|x_{<t}, y_{t-1})$$
$$p_\theta(x_t|x_{<t}, y_{\leq t}) = p_\theta(x_t|x_{<t}, y_t)$$

- Transition probability: Current label *conditionally independent* of all past labels given only previous label
- Emission probability: Current word *conditionally independent* of all past labels given only current label





- Is this a reasonable assumption for tagging? (Note that even if the assumption is false we can still use this model on any data.)
- But now marginalization and inference can be done exactly in time linear (rather than exponential) in sequence length.

Forward Algorithm for Exact Marginalization

- Now no need to consider all $|\mathcal{Y}|^T$ candidates because of the Markov assumptions
- This is a dynamic programming (DP) algorithm. Given $x_1 \ldots x_T$, the DP table we fill out is $\pi \in \mathbb{R}^{T \times |\mathcal{Y}|}$ where

$$\pi(t, y) = \sum_{y_1 \ldots y_t \in \mathcal{Y}^t:\ y_t = y} p_\theta(x_1 \ldots x_t, y_1 \ldots y_t)$$

- Output $\sum_{y \in \mathcal{Y}} \pi(T, y) \times p_\theta(y_* | x_{\leq T}, y)$ as the marginal probability of $x_1 \ldots x_T$
- We will see that computing each $\pi(t, y)$ will only take $O(|\mathcal{Y}|)$ time, hence the total runtime is $O(T\,|\mathcal{Y}|^2)$.
- Base case is easy: Compute for all $y \in \mathcal{Y}$

$$\pi(1, y) = p_\theta(y | y_0) \times p_\theta(x_1 | y)$$

Forward Algorithm: Main Body ($t > 1$)

$$
\begin{aligned}
\pi(t, y') &= \sum_{y_{\leq t}:\ y_t = y'} p_\theta(x_{\leq t}, y_{\leq t}) \\
&= \sum_{y_{<t}} p_\theta(x_{\leq t}, y_{<t}\ y') \\
&= \sum_{y_{<t}} p_\theta(x_{<t}, y_{<t}) \times p_\theta(y' | x_{<t}, y_{<t}) \times p_\theta(x_t | x_{<t}, y_{<t}, y')
\end{aligned}
$$

Forward Algorithm: Main Body ($t > 1$)

$$\begin{aligned}
\pi(t, y') &= \sum_{y_{\leq t}:\ y_t = y'} p_\theta(x_{\leq t}, y_{\leq t}) \\
&= \sum_{y_{<t}} p_\theta(x_{\leq t}, y_{<t}\ y') \\
&= \sum_{y_{<t}} p_\theta(x_{<t}, y_{<t}) \times p_\theta(y' | x_{<t}, y_{<t}) \times p_\theta(x_t | x_{<t}, y_{<t}, y') \\
&\stackrel{*}{=} \sum_{y_{<t}} p_\theta(x_{<t}, y_{<t}) \times p_\theta(y' | x_{<t}, y_{t-1}) \times p_\theta(x_t | x_{<t}, y')
\end{aligned}$$

Forward Algorithm: Main Body ($t > 1$)

$$
\begin{aligned}
\pi(t, y') &= \sum_{y_{\leq t}:\ y_t = y'} p_\theta(x_{\leq t}, y_{\leq t}) \\
&= \sum_{y_{<t}} p_\theta(x_{\leq t}, y_{<t}\ y') \\
&= \sum_{y_{<t}} p_\theta(x_{<t}, y_{<t}) \times p_\theta(y' | x_{<t}, y_{<t}) \times p_\theta(x_t | x_{<t}, y_{<t}, y') \\
&\stackrel{*}{=} \sum_{y_{<t}} p_\theta(x_{<t}, y_{<t}) \times p_\theta(y' | x_{<t}, y_{t-1}) \times p_\theta(x_t | x_{<t}, y') \\
&= \sum_{y} \sum_{y_{<t-1}} p_\theta(x_{<t}, y_{<t-1}\ y) \times p_\theta(y' | x_{<t}, y) \times p_\theta(x_t | x_{<t}, y')
\end{aligned}
$$

Forward Algorithm: Main Body ($t > 1$)

$$
\begin{aligned}
\pi(t, y') &= \sum_{y_{\leq t}:\ y_t = y'} p_\theta(x_{\leq t}, y_{\leq t}) \\
&= \sum_{y_{<t}} p_\theta(x_{\leq t}, y_{<t}\ y') \\
&= \sum_{y_{<t}} p_\theta(x_{<t}, y_{<t}) \times p_\theta(y' | x_{<t}, y_{<t}) \times p_\theta(x_t | x_{<t}, y_{<t}, y') \\
&\stackrel{*}{=} \sum_{y_{<t}} p_\theta(x_{<t}, y_{<t}) \times p_\theta(y' | x_{<t}, y_{t-1}) \times p_\theta(x_t | x_{<t}, y') \\
&= \sum_{y} \sum_{y_{<t-1}} p_\theta(x_{<t}, y_{<t-1}\ y) \times p_\theta(y' | x_{<t}, y) \times p_\theta(x_t | x_{<t}, y') \\
&= \sum_{y} \underbrace{\pi(t-1, y)}_{\text{already computed}} \times p_\theta(y' | x_{<t}, y) \times p_\theta(x_t | x_{<t}, y')
\end{aligned}
$$

Viterbi Algorithm for Exact Inference

- Same idea: No need to consider all $|\mathcal{Y}|^T$ candidates because of the Markov assumptions
- Given $x_1 \ldots x_T$, the DP table we fill out is $\pi \in \mathbb{R}^{T \times |\mathcal{Y}|}$ where

$$\pi(t, y) = \max_{y_1 \ldots y_t \in \mathcal{Y}^t \colon y_t = y} p_\theta(x_1 \ldots x_t, y_1 \ldots y_t)$$

- Exactly the same as forward if we **switch sum with max**

$$\pi(1, y) = p_\theta(y|y_0) \times p_\theta(x_1|y)$$
$$\pi(t, y') = \max_{y} \pi(t-1, y) \times p_\theta(y'|x_{<t}, y) \times p_\theta(x_t|x_{<t}, y')$$

- But this only gives us the joint probability of $x_1 \ldots x_T$ and its most likely tag sequence. How do we extract the actual tag sequence?

Backtracking for Viterbi

- Keep an additional chart to record the **path**:

$$\beta(t, y') = \arg\max_{y \in \mathcal{Y}} \ \pi(t-1, y) \times p_\theta(y' | x_{<t}, y) \times p_\theta(x_t | x_{<t}, y')$$

for $t = 2 \ldots T$.

- After running Viterbi, we can "backtrack"

$$\begin{aligned} y_T^* &= \arg\max_{y \in \mathcal{Y}} \ \pi(T, y) \times p_\theta(y_* | x_{\leq T}, y) \\ y_{T-1}^* &= \beta(T, y_T^*) \\ &\vdots \\ y_1^* &= \beta(2, y_2^*) \end{aligned}$$

and return $y_1^* \ldots y_T^*$.

Other Details

- In practice, we always operate in **log space** for numerical stability. The DP tables will store log probabilities, e.g., in forward

$$\pi(1, y) = \log p_\theta(y|y_0) + \log p_\theta(x_1|y)$$
$$\pi(t, y') = \operatorname*{logsumexp}_{y} \Big(\pi(t-1, y) + \log p_\theta(y'|x_{<t}, y) + \log p_\theta(x_t|x_{<t}, y')\Big)$$

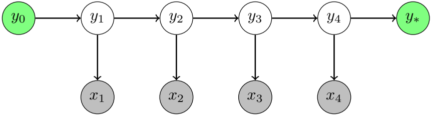
  where $\operatorname{logsumexp}_y f(y) = \log \sum_y \exp(f(y))$ is the usual numerically stable calculation for log space
- **Debugging.** Debugging is crucial, the first DP implementation is almost certainly incorrect.
  - Construct a small model randomly (e.g., with $|\mathcal{Y}| = 5$)
  - Generate a short sequence (e.g., $x_1 \dots x_7$) and compute marginalization and inference exactly by brute-force
  - Check if the output of forward/Viterbi matches with brute-force

The Hidden Markov Model

- Further Markov assumption on observation generation yields **hidden Markov model (HMM)**

$$p_\theta(y_t|x_{<t}, y_{<t}) = t_\theta(y_t|y_{t-1})$$
$$p_\theta(x_t|x_{<t}, y_{\leq t}) = o_\theta(x_t|y_t)$$

- Simplest form of labeled sequence generation



$$p_\theta(x_1 \dots x_T, y_1 \dots y_T) = \prod_{t=1}^{T} \underbrace{t_\theta(y_t|y_{t-1})}_{\text{transition prob}} \times \underbrace{o_\theta(x_t|y_t)}_{\text{emission prob}} \times t_\theta(y_*|y_T)$$

- Central model in NLP and machine learning: Tagging English text with a probabilistic model (Merialdo, 1994)
- Underlying tag sequence often unobserved (hence "hidden")

Forward Algorithm for HMMs in Matrix Form

- Organize HMM probabilities in matrix form
  - Emission matrix: $O \in \mathbb{R}^{|\mathcal{V}| \times |\mathcal{Y}|}$ where $O_{x,y} = o_\theta(x|y)$
  - Transition matrix: $T \in \mathbb{R}^{|\mathcal{Y}| \times |\mathcal{Y}|}$ where $T_{y',y} = t_\theta(y'|y)$
- Forward algorithm

$$p_\theta(x_1 \ldots x_T) = \underbrace{\tau_\infty^\top}_{1 \times |\mathcal{Y}|} \underbrace{\mathrm{diag}(O_{x_T})}_{|\mathcal{Y}| \times |\mathcal{Y}|} \underbrace{T}_{|\mathcal{Y}| \times |\mathcal{Y}|} \cdots \underbrace{\mathrm{diag}(O_{x_1})}_{|\mathcal{Y}| \times |\mathcal{Y}|} \underbrace{\tau_0}_{|\mathcal{Y}| \times 1}$$

$O_x \in \mathbb{R}^{|\mathcal{Y}|}$ is row x of O, $[\tau_0]_y = t_\theta(y|y_0)$, $[\tau_\infty]_y = t_\theta(y_*|y)$

- Compact/insightful view of stepwise marginalization in dynamic programming as matrix-matrix product

$$\sum_{y \in \mathcal{Y}} \pi(t-1, y) \times t_\theta(y'|y) \times o_\theta(x_t|y')$$

Learning HMMs

- **Supervised.** If $y_1 \dots y_T$ observed, just maximize

$$\log p_\theta(x_1 \dots x_T, y_1 \dots y_T) = \sum_{t=1}^{T} \log t_\theta(y_t | y_{t-1}) + \log o_\theta(x_t | y_t)$$

  Pre-neural: Parameters are *raw probabilities*, closed-form MLE by constrained optimization

$$t(y'|y) = \frac{\textbf{count}(y, y')}{\sum_{y' \in \mathcal{Y}} \textbf{count}(y, y')} \quad o(x|y) = \frac{\textbf{count}(x, y)}{\sum_{x \in \mathcal{V}} \textbf{count}(x, y)}$$

  (i.e., "training" means counting word/tag bigrams off of labeled sequences). If parametric, can do gradient ascent

- **Unsupervised.** If $y_1 \dots y_T$ unobserved, can still maximize *marginal* probability of $x_1 \dots x_T$

$$\log p_\theta(x_1 \dots x_T) = \log \underbrace{\sum_{y_1 \dots y_T} p_\theta(x_1 \dots x_T, y_1 \dots y_T)}_{\text{computable with forward alg.}}$$