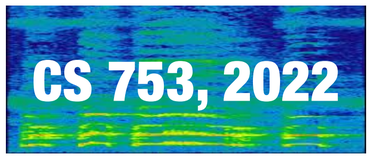
Live Session
(Pre-midsem)

Lecture 7a



Instructor: Preethi Jyothi, IITB

Question 1: WFSTs for ASR

Recall the WFST-based framework for ASR that was described in class. Given a test utterance x, let D_x be a WFST over the tropical semiring (with weights specialized to the given utterance) such that decoding the utterance corresponds to finding the shortest path in D_x. Suppose we modify D_x by adding γ (> 0) to each arc in D_x that emits a word. Let's call the resulting WFST D'_x.

A) Describe informally, what effect increasing γ would have on the word sequence obtained by decoding D'_x.

↑ing γ → result in decoded hyps being of smaller length

B) Recall that decoding D_x was used as an approximation for $\operatorname{argmax}_W \Pr(x|W) \Pr(W)$. What would be the analogous expression for decoding from D'_x?

$-\log \Pr(x|W) - \log \Pr(W) \; \boxed{+\gamma|W|}$

$\Pr(x|W) \Pr(W) \, e^{-\gamma|W|}$

Question 2: Morpheme-based ASR

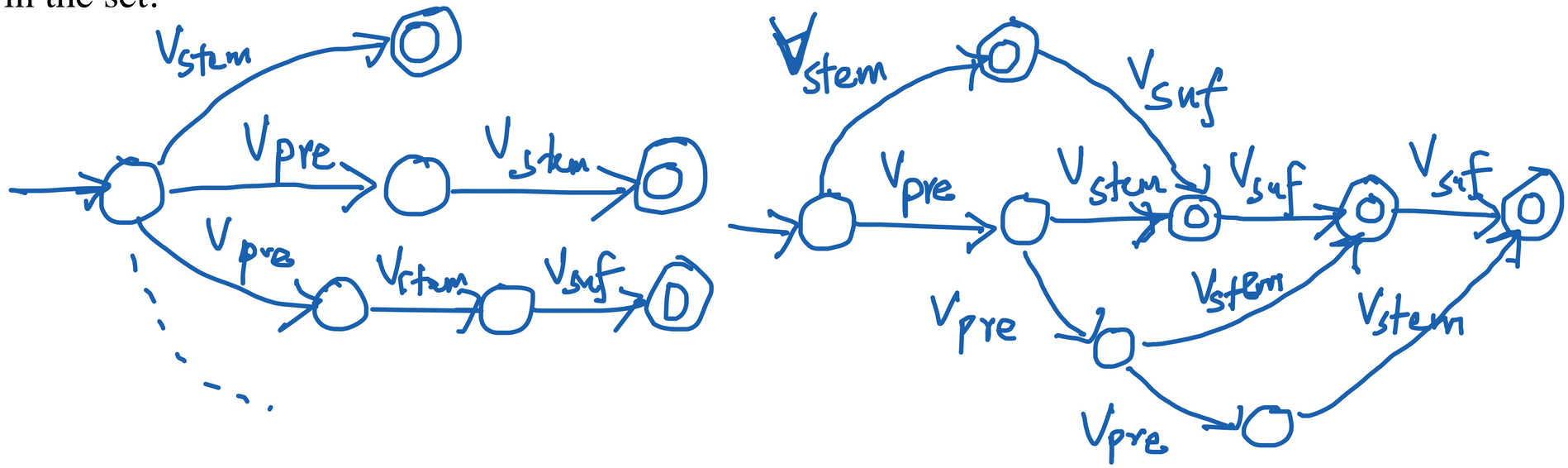
Words in a language can be composed of sub-word units called morphemes. For simplicity, in this problem, we consider there to be three sets of morphemes, $\mathrm{V}_{\mathrm{pre}}$,$\mathrm{V}_{\mathrm{stem}}$ and $\mathrm{V}_{\mathrm{suf}}$ – corresponding to prefixes, stems and suffixes. Further, we will assume that every word consists of a single stem, and zero or more prefixes and suffixes. That is, a word is of the form $w = p_1 \ldots p_k \sigma s_1 \ldots s_l$ where $k, l \geq 0$, and $p_i \in \mathrm{V}_{\mathrm{pre}}$, $s_i \in \mathrm{V}_{\mathrm{suf}}$ and $\sigma \in \mathrm{V}_{\mathrm{stem}}$. For example, a word like fair consists of a single morpheme (a stem), where as the word unfairness is composed of three morphemes, un + fair + ness, which are a prefix, a stem and a suffix, respectively.

A) Suppose we want to build an ASR system for a language using morphemes instead of words as the basic units of language. Which WFST(s) in the $\mathrm{H} \circ \mathrm{C} \circ \mathrm{L} \circ \mathrm{G}$ framework should be modified in order to utilize morphemes?

Question 2: Morpheme-based ASR

Words in a language can be composed of sub-word units called morphemes. For simplicity, in this problem, we consider there to be three sets of morphemes, V_{pre},V_{stem} and V_{suf} – corresponding to prefixes, stems and suffixes. Further, we will assume that every word consists of a single stem, and zero or more prefixes and suffixes. That is, a word is of the form $w = p_1 \ldots p_k \sigma s_1 \ldots s_l$ where $k, l \geq 0$, and $p_i \in V_{pre}$, $s_i \in V_{suf}$ and $\sigma \in V_{stem}$. For example, a word like fair consists of a single morpheme (a stem), where as the word unfairness is composed of three morphemes, un + fair + ness, which are a prefix, a stem and a suffix, respectively.

(B) Draw an FSA over morphemes ($V_{pre} \cup V_{stem} \cup V_{suf}$) that accepts only words with at most four morphemes. Your FSA should not have more than 10 states. You may draw a single arc labeled with a set to indicate a collection of arcs, each labeled with an element in the set.



Question 2: Morpheme-based ASR

Words in a language can be composed of sub-word units called morphemes. For simplicity, in this problem, we consider there to be three sets of morphemes, V_{pre},V_{stem} and V_{suf} – corresponding to prefixes, stems and suffixes. Further, we will assume that every word consists of a single stem, and zero or more prefixes and suffixes. That is, a word is of the form $w = p_1 \ldots p_k \sigma s_1 \ldots s_l$ where $k, l \geq 0$, and $p_i \in V_{pre}$, $s_i \in V_{suf}$ and $\sigma \in V_{stem}$. For example, a word like fair consists of a single morpheme (a stem), where as the word unfairness is composed of three morphemes, un + fair + ness, which are a prefix, a stem and a suffix, respectively.

(B) Draw an FSA over morphemes ($V_{pre} \cup V_{stem} \cup V_{suf}$) that accepts only words with at most four morphemes. Your FSA should not have more than 10 states. You may draw a single arc labeled with a set to indicate a collection of arcs, each labeled with an element in the set.

Question 3: Viterbi Variant

Modify the original Viterbi algorithm such that it returns the best state sequence among all sequences that never stay in a state for more than k consecutive transitions.

$V_t(j)$: Viterbi path probability

$$V_t(j) = \max_{d=1}^{k} V_{t,d}(j)$$

$$d > 1 : \quad V_{t,d}(j) = V_{t-1,d-1}(j)\, a_{jj}\, b_j(o_t)$$

$$d = 1 : \quad V_{t,1}(j) = \max_{i \neq j} \max_{d=1}^{k} V_{t-1,d}(i)\, a_{ij}\, b_j(o_t)$$

Question 4: CTC

The CTC training criterion is defined using a function $\mathcal{B}$ that maps a per-frame output sequence $\mathbf{a} = (a_1, ..., a_T)$ to a final output sequence $\mathbf{y} = (y_1, \ldots, y_N)$, by first compressing each run of an identical character in $\mathbf{a}$ to a run of length 1, and then removing all occurrences of the special blank symbol $\boxed{\epsilon}$. Here $y_j \in V$ and $a_i \in V \cup \{\boxed{\epsilon}\}$, where V is the output vocabulary.

Given an input sequence $\mathbf{x}$ of length T and an output character sequence $\mathbf{y}$ of length N, the CTC objective function is given by:

$$\Pr_{CTC}(\mathbf{y}|\mathbf{x}) = \sum_{\mathbf{a}:\mathcal{B}(\mathbf{a})=\mathbf{y}} \Pr(\mathbf{a}|\mathbf{x}).$$

Now suppose we would like to avoid the use of the blank symbol. Towards this we use a "contextualized" CTC. Here, each a_i is either a single symbol in V or a pair of symbols in V (a bigram). That is, the alphabet of a_i is now $W := V \cup V \times V$. We define $\mathcal{B}$ such that $\mathcal{B}(a_1, \ldots, a_T) = (y_1, \ldots, y_N)$ iff

$$\mathbf{a} = (\underbrace{y_1, \ldots, y_1}_{u_1 \text{times}}, \underbrace{(y_1, y_2), \ldots, (y_1, y_2)}_{b_1 \text{ times}}, \underbrace{y_2, \ldots, y_2}_{u_2 \text{ times}}, \underbrace{(y_2, y_3) \ldots, (y_2, y_3)}_{b_2 \text{ times}}, \ldots, \underbrace{(y_{N-1}, y_N), \ldots, (y_{N-1}, y_N)}_{b_{N-1} \text{times}}, \underbrace{y_N, \ldots, y_N}_{u_N \text{ times}})$$

where the "unigram" counts $u_i > 0$ for and all $i = 1, \ldots, N$ and the "bigram" counts $b_i > 0$ for all $i = 1, \ldots, N-1$ (if $N = 1$, no bigrams can be present). If $\mathbf{a}$ does not have this form, where two consecutive runs of unigrams are separated by a run of matching bigrams, then $\mathcal{B}(\mathbf{a}) = \perp$ (an error symbol). For example, $\mathcal{B}(a, a, (a, b), (a, b), b, (b, c), c, c) = (a, b, c)$. On the other hand, $\mathcal{B}(a, b, c) = \perp$, $\mathcal{B}(a, (a, b), b, (b, b), (b, c), c) = \perp$ and $\mathcal{B}(a, (a, b), (b, c), c) = \perp$.

Given a T frame input $\mathbf{x}$, suppose a neural network trained using CTC gives probabilities $\Pr(a|\mathbf{x}, t)$ for each $a \in W$ and $t = 1, \ldots, T$. Describe a dynamic programming algorithm which, given these probabilities and $\mathbf{y} \in V^N$ finds $\sum_{\mathbf{a}:\mathcal{B}(\mathbf{a})=\mathbf{y}} \Pr(\mathbf{a}|\mathbf{x})$ under the independence assumption used in CTC. (Defining a new recurrence for the CTC forward algorithm, as we've seen in class, will suffice.)

Question 4: CTC

The CTC training criterion is defined using a function $\mathcal{B}$ that maps a per-frame output sequence $\mathbf{a} = (a_1, ..., a_T)$ to a final output sequence $\mathbf{y} = (y_1, \ldots, y_N)$, by first compressing each run of an identical character in $\mathbf{a}$ to a run of length 1, and then removing all occurrences of the special blank symbol $\boxed{\epsilon}$. Here $y_j \in V$ and $a_i \in V \cup \{\boxed{\epsilon}\}$, where V is the output vocabulary.

Given an input sequence $\mathbf{x}$ of length T and an output character sequence $\mathbf{y}$ of length N, the CTC objective function is given by:

$$\Pr_{CTC}(\mathbf{y}|\mathbf{x}) = \sum_{\mathbf{a}:\mathcal{B}(\mathbf{a})=\mathbf{y}} \Pr(\mathbf{a}|\mathbf{x}).$$

Now suppose we would like to avoid the use of the blank symbol. Towards this we use a "contextualized" CTC. Here, each a_i is either a single symbol in V or a pair of symbols in V (a bigram). That is, the alphabet of a_i is now $W := V \cup V \times V$. We define $\mathcal{B}$ such that $\mathcal{B}(a_1, \ldots, a_T) = (y_1, \ldots, y_N)$ iff

$$\mathbf{a} = (\underbrace{y_1, \ldots, y_1}_{u_1 \text{times}}, \underbrace{(y_1, y_2), \ldots, (y_1, y_2)}_{b_1 \text{ times}}, \underbrace{y_2, \ldots, y_2}_{u_2 \text{ times}}, \underbrace{(y_2, y_3) \ldots, (y_2, y_3)}_{b_2 \text{ times}}, \ldots, \underbrace{(y_{N-1}, y_N), \ldots, (y_{N-1}, y_N)}_{b_{N-1} \text{times}}, \underbrace{y_N, \ldots, y_N}_{u_N \text{ times}})$$

where the "unigram" counts $u_i > 0$ for and all $i = 1, \ldots, N$ and the "bigram" counts $b_i > 0$ for all $i = 1, \ldots, N-1$ (if $N = 1$, no bigrams can be present). If $\mathbf{a}$ does not have this form, where two consecutive runs of unigrams are separated by a run of matching bigrams, then $\mathcal{B}(\mathbf{a}) = \perp$ (an error symbol). For example, $\mathcal{B}(a, a, (a, b), (a, b), b, (b, c), c, c) = (a, b, c)$. On the other hand, $\mathcal{B}(a, b, c) = \perp$, $\mathcal{B}(a, (a, b), b, (b, b), (b, c), c) = \perp$ and $\mathcal{B}(a, (a, b), (b, c), c) = \perp$.

Given a T frame input $\mathbf{x}$, suppose a neural network trained using CTC gives probabilities $\Pr(a|\mathbf{x}, t)$ for each $a \in W$ and $t = 1, \ldots, T$. Describe a dynamic programming algorithm which, given these probabilities and $\mathbf{y} \in V^N$ finds $\sum_{\mathbf{a}:\mathcal{B}(\mathbf{a})=\mathbf{y}} \Pr(\mathbf{a}|\mathbf{x})$ under the independence assumption used in CTC. (Defining a new recurrence for the CTC forward algorithm, as we've seen in class, will suffice.)

$$\alpha_1(1) = b_1(y_1')$$

[$b_t(y_j')$ is the probability given by the NN to the symbol y_j' when $|x| = T$ for $t = 1, \ldots, T$]

$$\alpha_t(j) = b_t(y_j') \left[\alpha_{t-1}(j-1) + \alpha_{t-1}(j) \right]$$

where

$$y_j' = \begin{cases} y_{\frac{j+1}{2}} & \text{if } j \text{ is odd } (j = 1, \ldots, 2N-1 \text{ when } |y| = N) \quad // \text{unigram} \\ (y_{\frac{j}{2}}, y_{\frac{j}{2}+1}) & \text{otherwise} \quad // \text{bigram} \end{cases}$$

$$\Pr_{CTC}(y|x) = \alpha_T(2N-1)$$

Question 5: Kneser-Ney Variant

Recall that a bigram language model with Kneser-Ney smoothing sets the probability of a word b to follow a word a as:

$$\Pr_{\mathrm{KN}}[b \mid a] := \frac{\max\{\pi(a,b) - d, 0\}}{\pi(a)} + \frac{d \cdot |\Psi(a)| \cdot |\Phi(b)|}{\pi(a) \cdot |B|}. \tag{1}$$

$\Pr(\epsilon \mid a) \cdot \Pr(b \mid \epsilon)$

Here $\pi(a,b)$ denotes the count of the bigram ab in the data and $\pi(a) = \sum_y \pi(a,y)$; the sets involved are $\Psi(a) := \{y \mid \pi(a,y) > 0\}$, $\Phi(b) := \{x \mid \pi(x,b) > 0\}$, and $B := \{(x,y) \mid \pi(x,y) > 0\}$. d is an "absolute discount" parameter that was subtracted from the counts $\pi(a,b)$. That is, it was assumed that d occurrences of ab in fact occurred as $a\epsilon b$ (i.e., the context was "forgotten" after producing a, and b was produced with an empty context), and the remaining $\pi(a,b) - d$ occurrences corresponded to the bigram.

(I) Instead of an absolute discounting d from the counts $\pi(a,b)$, one could use an adaptive discounting where each count $\pi(a,b)$ is discounted by $\mathcal{D}(\pi(a,b))$ where $\mathcal{D}$ is a discounting function such that $0 \leq \mathcal{D}(n) \leq n$. Then the first term in the expression for $\Pr_{\mathrm{KN}}[b \mid a]$ changes to

$$\frac{\pi(a,b) - \mathcal{D}(\pi(a,b))}{\pi(a)}.$$

What should the second term be in this case? Explain how you arrived at your answer.

$$\Pr(\epsilon \mid a)\Pr(b \mid \epsilon) = \frac{\sum_y \mathcal{D}(\pi(a,y))}{\pi(a)} \cdot \frac{\sum_x \mathcal{D}[\pi(x,b)]}{\sum_{x,y} \mathcal{D}[\pi(x,y)]}$$

Question 5: Kneser-Ney Variant

Recall that a bigram language model with Kneser-Ney smoothing sets the probability of a word b to follow a word a as:

$$\Pr_{\mathrm{KN}}[b \mid a] := \frac{\max\{\pi(a,b) - d, 0\}}{\pi(a)} + \frac{d \cdot |\Psi(a)| \cdot |\Phi(b)|}{\pi(a) \cdot |B|}. \tag{1}$$

Here $\pi(a,b)$ denotes the count of the bigram ab in the data and $\pi(a) = \sum_y \pi(a,y)$; the sets involved are $\Psi(a) := \{y \mid \pi(a,y) > 0\}$, $\Phi(b) := \{x \mid \pi(x,b) > 0\}$, and $B := \{(x,y) \mid \pi(x,y) > 0\}$. d is an "absolute discount" parameter that was subtracted from the counts $\pi(a,b)$. That is, it was assumed that d occurrences of ab in fact occurred as $a\epsilon b$ (i.e., the context was "forgotten" after producing a, and b was produced with an empty context), and the remaining $\pi(a,b) - d$ occurrences corresponded to the bigram.

(I)

Question 5: Kneser-Ney Variant

Recall that a bigram language model with Kneser-Ney smoothing sets the probability of a word b to follow a word a as:

$$\Pr_{\mathrm{KN}}[b \mid a] := \frac{\max\{\pi(a,b) - d, 0\}}{\pi(a)} + \frac{d \cdot |\Psi(a)| \cdot |\Phi(b)|}{\pi(a) \cdot |B|}. \tag{1}$$

Here $\pi(a,b)$ denotes the count of the bigram ab in the data and $\pi(a) = \sum_y \pi(a,y)$; the sets involved are $\Psi(a) := \{y \mid \pi(a,y) > 0\}$, $\Phi(b) := \{x \mid \pi(x,b) > 0\}$, and $B := \{(x,y) \mid \pi(x,y) > 0\}$. d is an "absolute discount" parameter that was subtracted from the counts $\pi(a,b)$. That is, it was assumed that d occurrences of ab in fact occurred as $a\epsilon b$ (i.e., the context was "forgotten" after producing a, and b was produced with an empty context), and the remaining $\pi(a,b) - d$ occurrences corresponded to the bigram.

(II) In the case of absolute discount from above, the discounting function is given by $\mathcal{D}(n) = \min\{d, n\}$. Then, your general expression from the previous part would simplify to the expression in (1) under a certain assumption on the counts. What is this assumption?

All bigrams x, y that have a non-zero count should appear at least d times

Question 5: Kneser-Ney Variant

Recall that a bigram language model with Kneser-Ney smoothing sets the probability of a word b to follow a word a as:

$$\Pr_{\text{KN}}[b \mid a] := \frac{\max\{\pi(a,b) - d, 0\}}{\pi(a)} + \frac{d \cdot |\Psi(a)| \cdot |\Phi(b)|}{\pi(a) \cdot |B|}. \tag{1}$$

Here $\pi(a,b)$ denotes the count of the bigram ab in the data and $\pi(a) = \sum_y \pi(a,y)$; the sets involved are $\Psi(a) := \{y \mid \pi(a,y) > 0\}$, $\Phi(b) := \{x \mid \pi(x,b) > 0\}$, and $B := \{(x,y) \mid \pi(x,y) > 0\}$. d is an "absolute discount" parameter that was subtracted from the counts $\pi(a,b)$. That is, it was assumed that d occurrences of ab in fact occurred as $a\epsilon b$ (i.e., the context was "forgotten" after producing a, and b was produced with an empty context), and the remaining $\pi(a,b) - d$ occurrences corresponded to the bigram.

(III) Simplify your expression for $\Pr_{\text{KN}}[b \mid a]$ from from part (i) above, when the discount function is defined as $\mathcal{D}(n) = \delta n$, where $\delta \in [0,1]$.

$$\Pr_{KN}[b|a] = \frac{\pi(a,b)(1-\delta)}{\pi(a)} + \frac{\delta \sum_x \pi(x,b)}{\sum_x \pi(x)}$$

Question 6: LM Perplexity

Consider a simple language with vocabulary $\{A, B, C\}$ and bigram probabilities as follows:

	A	B	C	$\langle/s\rangle$
A	0.5	0.01	0.4	0.09
B	0.6	0.01	0.3	0.09
C	0.7	0.01	0.2	0.09
$\langle s\rangle$	0.33	0.33	0.33	0.01

P[C|A]

P(A|<s>)

What is the perplexity for the sentence $\langle s\rangle CAB\langle/s\rangle$ in this model? While normalizing, you should omit $\langle s\rangle$ and $\langle/s\rangle$ in counting the total number of words.

You need not evaluate the expression numerically.

$$-\frac{1}{3}\left(\log_2 0.33 + \log_2(0.7) + \log_2 0.01 + \log_2 0.09\right)$$

$$2^{\left(\frac{1}{Pr[\langle s\rangle CAB\langle/s\rangle]}\right)^{1/3}}$$

Question 6: LM Perplexity

Consider a simple language with vocabulary $\{A, B, C\}$ and bigram probabilities as follows:

	A	B	C	$\langle/s\rangle$
A	0.5	0.01	0.4	0.09
B	0.6	0.01	0.3	0.09
C	0.7	0.01	0.2	0.09
$\langle s\rangle$	0.33	0.33	0.33	0.01

Find a sentence obtained by applying a single substitution to the above sentence $\langle s\rangle CAB\langle/s\rangle$ (and not changing $\langle s\rangle$ or $\langle/s\rangle$) such that the resulting sentence has the minimum possible perplexity. Show your work.

<s>, x where x ∈ {A, B, C}

Keep C intact ∵ bigram prob of C, A is highest

Bigram prob of A, x is the highest for x = A

<s> C A A </s>