SINGAPORE UNIVERSITY OF TECHNOLOGY AND DESIGN

50.040 Natural Language Processing, Summer 2020
Homework 5

Due 17 Aug 2020, 5pm

Problem 1: Phrase-based Machine Translation (13 Points)

Consider the task of translating from Chinese into English using a phrase-based translation model. We have the following Chinese sentence (each Chinese word is in Pinyin):

`pingjia chaoshi shi xinjiapo zuidade chaoshi`

And we hope to have it translated into the following English sentence:

`Fairprice is the biggest supermarket in Singapore`

Assume we have the following translation rules (each comes with an ID in the brackets):

$$\texttt{pingjia} \rightarrow \texttt{Fairprice} \quad (1)$$
$$\texttt{pingjia chaoshi} \rightarrow \texttt{Fairprice} \quad (2)$$
$$\texttt{pingjia chaoshi shi} \rightarrow \texttt{Fairprice is the} \quad (3)$$
$$\texttt{shi} \rightarrow \texttt{is} \quad (4)$$
$$\texttt{xinjiapo} \rightarrow \texttt{Singapore} \quad (5)$$
$$\texttt{xinjiapo} \rightarrow \texttt{in Singapore} \quad (6)$$
$$\texttt{zuidade} \rightarrow \texttt{the biggest} \quad (7)$$
$$\texttt{zuidade} \rightarrow \texttt{biggest} \quad (8)$$
$$\texttt{chaoshi} \rightarrow \texttt{supermarket} \quad (9)$$

(a) Assume we would like to make use of the above phrase pairs to perform phrase-based translation. Provide one possible way of using the above phrase pairs to translate the source Chinese sentence into the English sentence *exactly*. Please provide the sequence of rule IDs (e.g., 1, 3, 7, ...) that shall be used during the translation process. Please list down the IDs of rules *strictly* in the order they shall be used in the translation process. *(4 points)*

Answer: There are two possible sequences. Either is considered correct.
Sequence 1: (2), (4), (7), (9), (6)
Sequence 2: (3), (8), (9), (6)

(b) As we have discussed during class, the distortion model can be calculated as follows:

$$\eta \times |\text{pl}(p_k) + 1 - \text{pf}(p_{k+1})|$$

where $\text{pl}(p_k)$ is the position of the last word in the Chinese phrase that corresponds to the previous translated English phrase, while $\text{pf}(p_{k+1})$ is the position of the first word in the Chinese phrase that corresponds to the current translated English phrase.

What is the score for the distortion model involved for the translation process in (a)? Write down the score in terms of η (i.e., the answer should be in the form of $\eta \times c$ where c is an integer). Clearly write down the steps that lead to your answer. *(3 points)*

Answer: Sequence 1: (2), (4), (7), (9), (6)

(2): `pingjia chaoshi` → `Fairprice`

$$\eta \times |\text{pl}(p_k) + 1 - \text{pf}(p_{k+1})| = |0 + 1 - 1| = 0$$

(4): `shi`→ `is`

$$\eta \times |\text{pl}(p_k) + 1 - \text{pf}(p_{k+1})| = |2 + 1 - 3| = 0$$

(7): `zuidade`→ `the biggest`

$$\eta \times |\text{pl}(p_k) + 1 - \text{pf}(p_{k+1})| = |3 + 1 - 5| = 1$$

(9): `chaoshi`→ `supermarket`

$$\eta \times |\text{pl}(p_k) + 1 - \text{pf}(p_{k+1})| = |5 + 1 - 6| = 0$$

(6): `xinjiapo`→ `in Singapore`

$$\eta \times |\text{pl}(p_k) + 1 - \text{pf}(p_{k+1})| = |6 + 1 - 4| = 3$$

$$c = 0 + 0 + 1 + 0 + 3 = 4$$

The answer is $\eta \times 4$

Sequence 2: (3), (8), (9), (6)

(3): `pingjia chaoshi shi` → `Fairprice is the`

$$\eta \times |\text{pl}(p_k) + 1 - \text{pf}(p_{k+1})| = |0 + 1 - 1| = 0$$

(8): `zuidade`→ `biggest`

$$\eta \times |\text{pl}(p_k) + 1 - \text{pf}(p_{k+1})| = |3 + 1 - 5| = 1$$

(9): `chaoshi`→ `supermarket`

$$\eta \times |\text{pl}(p_k) + 1 - \text{pf}(p_{k+1})| = |5 + 1 - 6| = 0$$

(6): `xinjiapo`→ `in Singapore`

$$\eta \times |\text{pl}(p_k) + 1 - \text{pf}(p_{k+1})| = |6 + 1 - 4| = 3$$

$$c = 0 + 0 + 1 + 0 + 3 = 4$$

The answer is $\eta \times 4$

(c) The phrase-based translation model also needs the third component, which is the language model. Assume we consider a bigram language model here. Write down the formula for the language model score for the translation process in (a). Feel free to use * to denote the special start symbol. *(1 points)*

Answer:

$$\log q(\text{Fairprice}|*) + \log q(\text{is}|\text{Fairprice}) + \log q(\text{the}|\text{is}) + \log q(\text{biggest}|\text{the})$$
$$+ \log q(\text{supermarket}|\text{biggest}) + \log q(\text{in}|\text{supermarket}) + \log q(\text{Singapore}|\text{in})$$

(d) Again, consider translating from the following Chinese sentence into the English sentence:

`pingjia chaoshi shi xinjiapo zuidade chaoshi`

`Fairprice is the biggest supermarket in Singapore`

Consider the following new set of rules, each of which is now associated with a weight:

`pingjia` → `Fairprice`	(1.0)	(1)
`pingjia chaoshi` → `Fairprice`	(−0.5)	(2)
`pingjia chaoshi shi` → `Fairprice is the`	(−1.0)	(3)
`shi` → `is`	(2.0)	(4)
`xinjiapo` → `Singapore`	(3.0)	(5)
`xinjiapo` → `in Singapore`	(0.5)	(6)
`zuidade` → `the biggest`	(2.0)	(7)
`zuidade` → `biggest`	(1.5)	(8)
`chaoshi` → `supermarket`	(1.0)	(9)

As we discussed in class, the score of a derivation D is calculated as follows:

$$\text{score}(D) = \text{score}_{LM}(D) + \text{score}_{TM}(D) + \text{score}_{DM}(D)$$

where score_{LM} is the log of the language model probability of the target translated English sentence, score_{TM} is the translation score, which is the sum of the weights of all the involved translation rules (e.g., +1.0 is the weight for the first rule above), and score_{DM} is the score of the distortion model as we discussed in class (which involves η).

Assume $\eta = -8.0$. Please provide the *optimal* sequence of rule IDs (e.g., 1, 3, 7, ...) that shall be used during the translation process (i.e., the derivation with the highest overall score). Please list down the IDs *strictly* in the order they shall be used in the translation process. **Clearly explain your answer.** (*Hints: for this question, do you need to know the exact language model scores?*) *(4 points)*

Answer: There are two possible sequences. Since for both sequences, the target sentences are the same, the language model scores for both cases are the same.

For sequence (1):

$$score_{TM}(D) = -0.5 + 2 + 2 + 1 + 0.5 = 5$$
$$score_{DM}(D) = \eta \times 4 = -32 \tag{10}$$
$$score(D) = score_{LM}(D) + 5 - 32 = score_{LM}(D) - 27 \tag{11}$$

For sequence (2):

$$score_{TM}(D) = -1 + 1.5 + 1 + 0.5 = 2$$

$$score_{DM}(D) = \eta \times 4 = -32 \tag{12}$$

$$score(D) = score_{LM}(D) + 2 - 32 = score_{LM}(D) - 30 \tag{13}$$

The score of sequence 1 is larger. Therefore, the optimal sequence of the rule IDs is (2)(4)(7)(9)(6)

(e) Now, assume we would like to revise the scoring function of a derivation D as follows:

$$\text{score}(D) = \text{score}_{LM}(D) + \text{score}_{TM}(D) + \text{score}_{DM}(D) + \text{penalty}(D)$$

In other words, on top of the original score, we would like to add a "penalty" to each derivation. The penalty is defined as follows:

$$\text{penalty}(D) = -|D|$$

where $|D|$ is the number of rules involved in the derivation D. For example, if the derivation D involves 7 rules, the penalty for this derivation is -7.

Now you would like to switch to the new scoring function when scoring a derivation. However you have already implemented your search algorithm for your decoder based on the old scoring function used in question (d), and you do not want to re-write your existing search algorithm. Think about how to make minimal changes to your implementation so that you can decode under the new scoring function. **Describe your solution clearly and concisely.** *(1 points)*

Answer: As the penalty is equal to the additive inverse of the number of rules, an alternative view is that each rule contributes a penalty of -1. Therefore for any rule with weight w, we change its weight to $w - 1$. No other changes would be required.

Problem 2: Synchronous Context Free Grammars (16 Points)

The synchronous CFG (SCFG) is able to parse two sentences in two different languages simultaneously. Consider the following synchronous grammar rules:

$$\mathbf{S} \rightarrow (\mathbf{X}_1,\ \mathbf{X}_1) \tag{1}$$

$$\mathbf{X} \rightarrow (\mathbf{X}_1\ \mathbf{X}_2,\ \mathbf{X}_1\ \mathbf{X}_2) \tag{2}$$

$$\mathbf{X} \rightarrow (\mathbf{X}_1\ \mathbf{X}_2,\ \mathbf{X}_2\ \mathbf{X}_1) \tag{3}$$

$$\mathbf{X} \rightarrow (\texttt{a},\ \texttt{A}) \tag{4}$$

$$\mathbf{X} \rightarrow (\texttt{b},\ \texttt{B}) \tag{5}$$

$$\mathbf{X} \rightarrow (\texttt{c},\ \texttt{C}) \tag{6}$$

$$\mathbf{X} \rightarrow (\texttt{d},\ \texttt{D}) \tag{7}$$

where **S** is the designated initial non-terminal (i.e., every derivation should start with this special non-terminal).

(a) Now, assume we would like to translate the following sentence:

`a b c d`

Using the above rules, we should be able to translate the above sequence into another sequence which consists of 4 target words from `A, B, C` and `D`. Note that each possible translation is essentially a possible permutation of the words `A, B, C` and `D` (i.e., each of these 4 words appears exactly once). For example, `A B C D` and `A B D C` are possible permutations/translations. There are totally 24 possible permutations for these 4 words. However, among these 24 permutations, 2 of them are not possible (i.e., it is not possible to use these rules to translate the above input sequence into a sequence consisting of `A, B, C` and `D` that follow a particular order). List down these 2 impossible permutations. *(2 points)*

Answer:

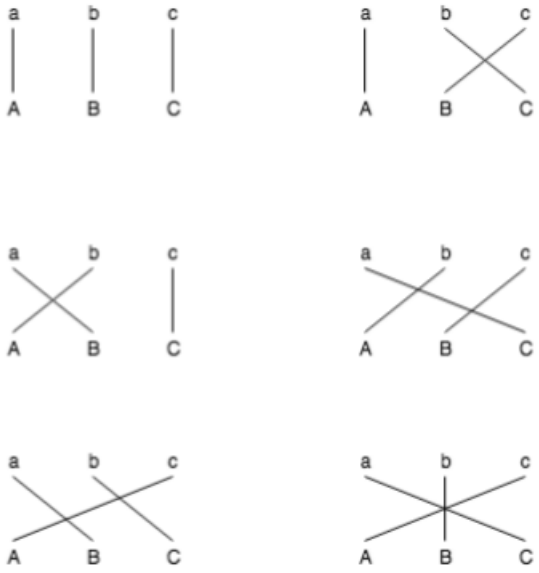
(b) Now, assume we would like to translate the following sentence:

`a b c`

List down all the possible translations (e.g., `A B C` may be one possible translation), using the above rules only. *(3 points)*

Answer: As shown below, the possible translations are:

1. A B C
2. A C B
3. B A C
4. C A B
5. B C A
6. C B A



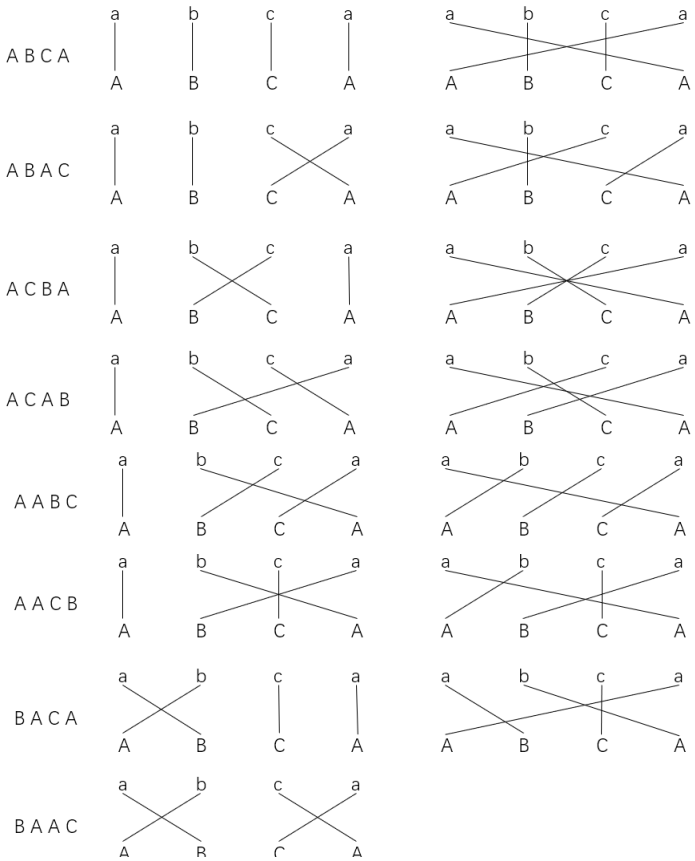
(c) Now, assume we would like to translate the following sentence:

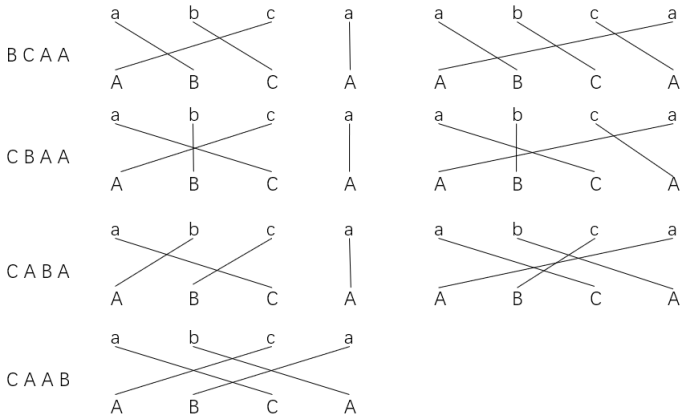
`a b c a`

How many possible translation can you have? List down all the possible translations (e.g., `A B C A` may be one possible translation), using the above rules only. *(6 points)*

Answer:

1. A B C A
2. A B A C
3. A C B A
4. A C A B
5. A A B C
6. A A C B
7. B A C A
8. B A A C
9. B C A A
10. C B A A
11. C A B A
12. C A A B





(d) Now we add a new synchronous grammar rule:

$$\mathbf{X} \rightarrow (\texttt{e},\ \texttt{E}) \tag{8}$$

We want to translate the new sentence `a b c d e` using the above 8 rules. Note that each possible translation is essentially a possible permutation of the words `A B C D E`. For example, `A B C D E` is one possible permutation (translation). Can we have odd number of impossible permutations in this case? Please explain your answer. *(5 points)*

Answer: The number of impossible permutations will always be an even number due to the symmetry of $(a, b, c, a), (A, B, C, A)$ (see the answer to question (a)).

Problem 3: Word Alignment Model (11 Points)

(a) IBM Model 1 discussed in class is a simple word alignment model that learns alignments between words from a pair of source and target sentences. The model requires "parallel data" as training set to automatically learn the alignment information from data. However, sometimes, instances (each instance is a sentence pair) from the training set may come from various different sources. Some sources may be more reliable while some may be less so (e.g., there may be incorrect translations for instances from unreliable sources). Although more data (even though the data may be less reliable) may be helpful, learning from unreliable data sources may also lead to sub-optimal results. To remedy this issue, we may use the following trick: we can create k (e.g., k=10, or k=100) copies of the training instances from the reliable source and add them into the training set before running the word alignment model. Doing so allows the algorithm to concentrate more on the reliable instances. However, on the other hand, doing so may make the learning process much longer as the training set now becomes much larger. Think of an alternative but equivalent way to do the same trick, without the need to explicitly create k copies of the training instances from the reliable source. *(Hint: can we assign a "weight" to each instance instead? If so, how shall we change the E and M steps?) (5 points)*

Answer: Let us use w_l to denote the weight of the l-th instance, and use (i, j) to denote an alignment between the i-th target word and j-th source word. If the instance is from the reliable source, set w_l to the large value k (e.g., 10 or 100); otherwise, set w_l to 1.

E step

- Soft EM:

$$\text{count}^{(l)}(i, j) = w_l \times \frac{t(f_i|e_j)}{\sum_{j'}^{n} t(f_i|e_{j'})}$$

- Hard EM:

$$\text{count}^{(l)}(i,j) = \begin{cases} w_l, & \text{if } j = \arg\max_{j'} t(f_i|e_{j'}) \\ 0, & \text{o.w.} \end{cases}$$

M step

$$t(f|e) = \frac{\text{count}(e,f)}{\text{count}(e)}$$

where:

$$count(e,f) = \sum_{l,f_i=f,e_j=e} \text{count}^{(l)}(i,j)$$

$$\text{count}(e) = \sum_{l,i,e_j=e} \text{count}^{(l)}(i,j)$$

(b) Consider finding the word alignments between Chinese and English. Sometimes, we may have some prior knowledge about how the words from these two different languages shall be aligned. For example, the two commas (,) from both Chinese and English shall be aligned with a high probability. Such prior knowledge can be used to improve the word alignment performance. Similarly, there may also be some other prior knowledge that can be exploited for improving the learning of word alignments. For example, we may already know the word-word correspondence between the following two words: "`daxue` – `university`". Typically, such information can be all stored into a lexicon in the form analogous to a Chinese-English dictionary, which consists of word-word translation pairs. Let us call this lexicon a "Chinese-English Dictionary". Now, assume you have already implemented IBM model 1, and there is a parallel corpus and a Chinese-English Dictionary available to you. Think of a convenient way to incorporate the Chinese-English Dictionary into the word alignment learning process. **Clearly explain your answer.** *(Hints: can we borrow some idea from question (a)?) (2 points)*

Answer: Create one instance for each entry in the Chinese-English Dictionary. Regard each such newly created instance as a reliable instance. Follow the procedure in (a).

(c) When training IBM Model 2, we need to randomly initialize the alignment and translation parameters q, t before applying Soft/Hard EM algorithm. However, the EM algorithm for IBM Model 2 may converge to different parameter estimates, depending on the initial parameters. Can you think of a better way to initialize q or t? *(4 points)*

Answer:

Step 1: Estimate the t parameters using the EM algorithm for IBM Model 1

Step 2: Use the $t(f|e)$ parameters estimated under IBM Model 1, in step 1; Use random values for the $q(j|i,l,m)$ parameters.

Problem 4: Attention (10 Points)

(a) During class, we have discussed several types of attention mechanisms. One of them is the "concatenation" attention mechanism. In one of the slides, it is written as follows:

$$\alpha_i = v^T \tanh(W[e_j; d_i])$$

where $[e_j; d_i]$ refers to the concatenation of the two column vectors e_j and d_i, and $e_j, d_i \in \mathbb{R}^k$. The concatenated vector $[e_j; d_i] \in \mathbb{R}^{2k}$, where k is a positive integer.

However, in another slide, it is written as follows:

$$\alpha_i = v^T \tanh(W_1 e_j + W_2 d_i)$$

Here W, W_1, and W_2 are all learnable matrices, and v is a learnable vector. In other words, they consist of parameters which need to be learned.

During class, it was mentioned that the above two different ways of writing the concatenation attention mechanism are essentially equivalent. Formally show why that is the case. *(4 points)*

Answer:

Assume $W \in \mathbb{R}^{d \times 2k}$. Then there exists two matrices $W_1^{'}, W_2^{'} \in \mathbb{R}^{d \times k}$, such that $W \equiv [W_1^{'}; W_2^{'}]$.

Now, we have:

$$W[e_j; d_i] = [W_1^{'}; W_2^{'}][e_j; d_i] = W_1^{'} e_j + W_2^{'} d_i$$

We can simply set $W_1 = W_1^{'}$ and $W_2 = W_2^{'}$. From here we can see for any W, we can find a pair W_1 and W_2 that can be used in the second formula that returns the same results.

Similarly, for any $W_1, W_2 \in \mathbb{R}^{d \times k}$, we can construct $W^{'} \in \mathbb{R}^{d \times 2k}$ as follows:

$$W^{'} = [W_1; W_2]$$

We can verify that:

$$W^{'}[e_j; d_i] = [W_1; W_2][e_j; d_i] = W_1 e_j + W_2 d_i$$

We can now simply set $W = W^{'}$. Again we can see that for any W_1 and W_2, we can find a W that can be used in the first formula that returns the same results. Thus, the two ways of writing the concatenation attention are essentially equivalent.

(b) After computing the attention scores $\boldsymbol{\alpha} = (\alpha_1, ..., \alpha_T)$, we need to normalize the attention scores using `softmax` function, obtaining an attention probability distribution $\boldsymbol{a}$.

$$a_i = \frac{e^{\alpha_i}}{\sum_j^T e^{\alpha_j}}, \quad i = 1, 2, ..., T$$

In practice, `softmax` operation can be numerically unstable when the values in $\boldsymbol{\alpha}$ are very negative or very positive, leading to `underflow` and `overflow` problems. `underflow/overflow` occur when the sum in the above equation is approximated as $0/\infty$, leaving the probability distribution undefined. One way to ensure numerical stability is to subtract $\max(\boldsymbol{\alpha})$ from every element in $\boldsymbol{\alpha}$ before we carry out `softmax` operation.

Please clearly show that this approach is equivalent to the above equation. Explain how does this approach address `overflow` and `underflow` problem. *(6 points)*

Answer: Let $\alpha_{max} = \max(\boldsymbol{\alpha})$ and $\boldsymbol{\alpha}^* = (\alpha_1, \alpha_2, ..., \alpha_T) - (\alpha_{max}, \alpha_{max}, ..., \alpha_{max})$ After normalization, the new attention probability distribution $\boldsymbol{a}^*$ will be

$$\begin{aligned} a_i^* = \frac{e^{\alpha_i - \alpha_{max}}}{\sum_j^T e^{\alpha_j - \alpha_{max}}} &= \frac{e^{-\alpha_{max}} \cdot e^{\alpha_i}}{e^{-\alpha_{max}} \cdot \sum_j^T e^{\alpha_j}} \\ &= \frac{e^{\alpha_i}}{\sum_j^T e^{\alpha_j}} = a_i, \quad i = 1, 2, ..., T \end{aligned}$$

Thus, these two normalization approaches are equivalent.

Subtracting $\max(\boldsymbol{\alpha})$ results in the largest argument to exp being 0, which rules out the possibility of overflow. Likewise, at least one term in the denominator has a value of 1, which rules out the possibility of underflow in the denominator leading to a division by zero.