13 Symbolic MT 3: Phrase-based MT

The previous two sections introduced word-by-word models of translation, how to learn them, and how to perform search with them. In this section, we'll discuss expansions of this method to **phrase-based machine translation** (PBMT; [7]), which uses "phrases" of multiple symbols, which have allowed for highly effective models in a number of sequence-to-sequence tasks.

13.1 Advantages of Memorizing Multi-symbol Phrases

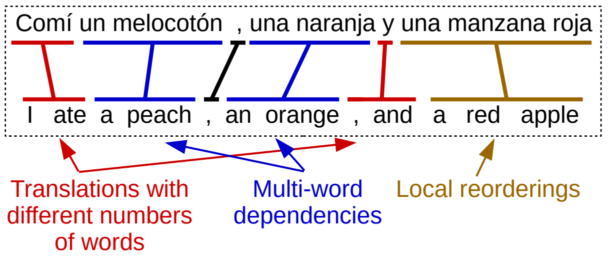
Comí un melocotón , una naranja y una manzana roja
I ate a peach , an orange , and a red apple
Translations with different numbers of words
Multi-word dependencies
Local reorderings

Figure 40: An example of a translation with phrases.

The basic idea behind PBMT is that we have a model that memorizes multi-symbol strings, and translates this string as a single segment. An example of what this would look like for Spanish-to-English translation is shown in the upper part of Figure 40, where each phrase in the source and target languages is underlined and connected by a line. From this example, we can observe a number of situations in which translation of phrases is useful:

Consistent Translation of Multiple Symbols: The first advantage of a phrase-based model is that it can memorize coherent units to ensure that all relevant words get translated in a consistent fashion. For example, the determiners "un" and "una" in Spanish can both be translated into either "an" or "a" in English. If these are translated separately from their accompanying noun, there is a good chance that the language model will help us choose the right one, but there is still significant room for error.[37] However, if we have phrases that "un melocotón" is almost always translated into "*a* peach" and "una naranja" is almost always translated into "*an* orange", these sorts of mistakes will be much less likely.

This is particularly true when translating technical phrases or proper names. For example, "Ensayo de un Crimen" is a title of a Mexican movie that translates literally into something like "an attempted crime". It is only if we essentially memorize this multi-word unit that we will be able to generate its true title, "The Criminal Life of Archibaldo de la Cruz".

Handling of Many-to-One Translations: Phrase-based models are also useful when handling translations where multiple words are translated into a single one or vice-versa. For example, in the example the single word "comió" is translated into "I ate". While word-based models have methods for generating words such as "I" from thin air, it is generally safer to remember this as a chunk and generate multiple words together from a single word, which is something that phrase-based models can do naturally.

[37]For example, the choice of "a" or "an" is not only affected by the probability of the following word $P(e_t \mid e_{t-1} = \text{“a/an''})$, which will be affected by whether e_t is a particular type of noun, but also the language model probability of "a" or "an" given the previous word $P(e_t = \text{“a/an''} \mid e_{t-1})$, which might randomly favor one or the other based on quirks in the statistics of the training corpus.

Handling of Local Re-ordering Finally, in addition to getting the translations of words correct, it is necessary to ensure that they get translated in the proper order. Phrase-based models also have some capacity for short-distance re-ordering built directly into the model by memorizing chunks that contain reordered words. For example, in the phrase translating "una manzana roja" to "a red apple", the order of "manazana/apple" and "roja/red" is reversed in the two languages. While this reordering between words can be modeled using an explicit reordering model (as described in Section 13.4), this can also be complicated and error prone. Thus, memorizing common reordered phrases can often be an effective for short-length reordering phenomena.

13.2 A Monotonic Phrase-based Translation Model

So now that it's clear that we would like to be modeling multi-word phrases, how do we express this in a formalized framework? First, we'll tackle the simpler case where there is no explicit reordering, which is also called the case of **monotonic** transductions.

In the previous section, we discussed an extremely simple monotonic model that modeled the noisy-channel translation model probability $P(F \mid E)$ in a word-to-word fashion as follows:

$$P(F \mid E) = \prod_{t=1}^{|E|} P(f_t \mid e_t). \tag{126}$$

To extend this model, we will first define $\bar{F} = \boldsymbol{f}_1, \ldots, \boldsymbol{f}_{|\bar{F}|}$ and $\bar{E} = \boldsymbol{e}_1, \ldots, \boldsymbol{e}_{|\bar{E}|}$, which are sequences of phrases. In the above example, this would mean that:

$$\bar{F} = \{\text{“comí”}, \text{“un melocotón”}, \ldots, \text{“una manzana roja”}\} \tag{127}$$

$$\bar{E} = \{\text{“i ate”}, \text{“a peach”}, \ldots, \text{“a red apple”}\}. \tag{128}$$

Given these equations, we would like to re-define our probability model $P(F \mid E)$ with respect to these phrases. To do so, assume sequential process where we first translate target words E into target phrases $\bar{E}$, then translate target phrases $\bar{E}$ into source phrases $\bar{F}$, then translate source phrases $\bar{F}$ into target words F. Assuming that all of these steps are independent, this can be expressed in the following equations:

$$P(F, \bar{F}, \bar{E} \mid E) = P(F \mid \bar{F}) P(\bar{F} \mid \bar{E}) P(\bar{E} \mid E). \tag{129}$$

Starting from the easiest sub-model first, $P(F \mid \bar{F})$ is trivial. This probability will be one whenever, the words in all the phrases of $\bar{F}$ can be concatenated together to form F, and zero otherwise. To express this formally, we define the following function

$$F = \text{concat}(\bar{F}). \tag{130}$$

The probability $P(\bar{E} \mid E)$, on the other hand is slightly less trivial. While $E = \text{concat}(\bar{E})$ must hold, there are multiple possible segmentations $\bar{E}$ for any particular E, and thus this probability is not one. There are a number of ways of estimating this probability, the most common being either a constant probability for all segmentations:

$$P(\bar{E} \mid E) = \frac{1}{Z}, \tag{131}$$

or a probability proportional to the number of phrases in the translation

$$P(\bar{E} \mid E) = \frac{|E|^{\lambda_{\text{phrase-penalty}}}}{Z}. \tag{132}$$

Here Z is a normalization constant that ensures that the probability sums to one over all the possible segmentations. The latter method has a parameter $\lambda_{\text{phrase-penalty}}$, which has the intuitive effect of controlling whether we attempt to use fewer longer phrases $\lambda_{\text{phrase-penalty}} < 0$ or more shorter phrases $\lambda_{\text{phrase-penalty}} > 0$. This penalty is often tuned as a parameter of the model, as explained in detail in Section 13.6.

Finally, the **phrase translation model** $P(\bar{F} \mid \bar{E})$ can is calculated in a way very similar to the word-based model, assuming that each phrase is independent:

$$P(\bar{F} \mid \bar{E}) = \prod_{t=1}^{|\bar{E}|} P(\bar{\boldsymbol{f}}_t \mid \bar{\boldsymbol{e}}_t). \tag{133}$$

This is conceptually simple, but it is necessary to be able to estimate the phrase translation probabilities $P(\bar{\boldsymbol{f}}_t \mid \bar{\boldsymbol{e}}_t)$ from data. We will describe thiss process in Section 13.3.

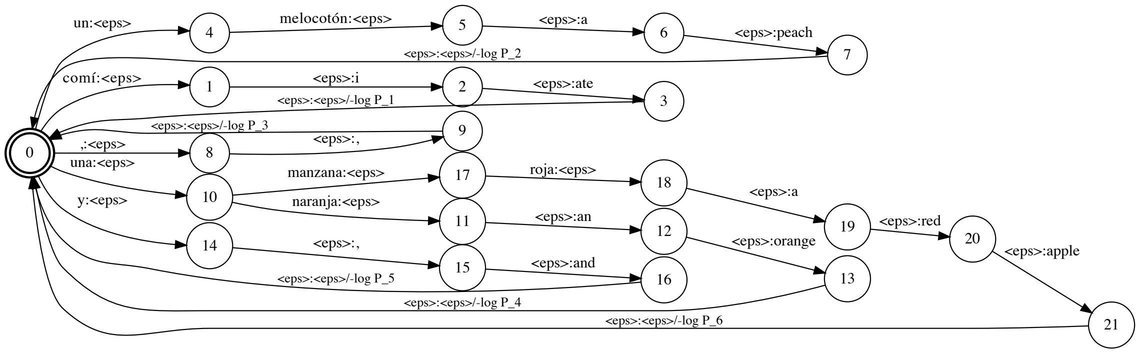

Figure 41: An example of an WFST for a phrase-based translation model. $-\log P\_n$ is an abbreviation for the negative log probability of the nth phrase, i.e., $-\log P\_1$ is equal to $P(\bar{\boldsymbol{f}} = \text{“comí”} \mid \bar{\boldsymbol{e}} = \text{“i ate”})$.

But first, a quick note on how we would express a phrase-based translation model as a WFST. One of the nice things about the WFST framework is that this is actually quite simple; we simply create a path through the WFST that:

1. First reads in source words one at a time.
2. Then prints out the target words one at a time.
3. Finally, adds the log probability.

An example of this (using the phrases from Figure 40) is shown in Figure 41. This model can be essentially plugged in instead of the word-based translation model used in Section 12.

13.3 Phrase Extraction and Scoring

So now we have a translation model, how do we extract phrases from data and calculate their translation scores? Basically, the intuition behind the method is that we want to extract phrases that are *consistent* with word alignments, such as those obtained by the IBM models introduced in Section 11.

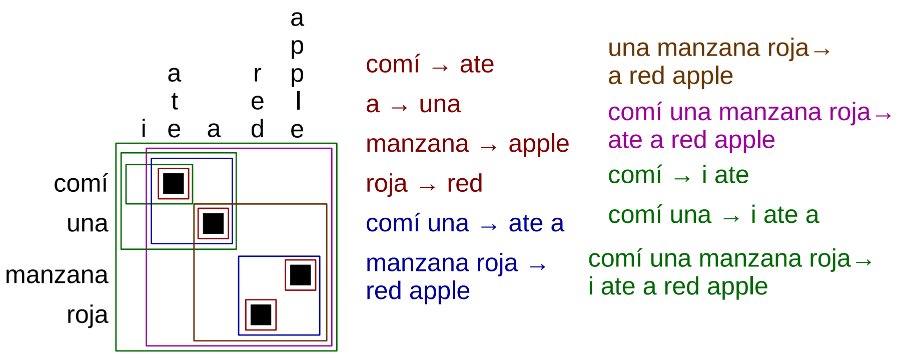

Figure 42: Phrase extraction from aligned data. Phrases of various lengths (1: red, 2: blue, 3: yellow, 4: purple) are extracted. Phrases containing the null-aligned word "i" are extracted by appending it to neighboring phrases (green).

An example of such alignments, and the phrases extracted from them, are shown in Figure 42. In the example, we can note a few things. First, phrases of various lengths are extracted, from short phrases containing word-to-word alignments, and long phrases containing an entire sentence.[38] This is important because it allows the translation system to remember and use longer phrases to improve its modeling accuracy, but also makes it possible to fall back to shorter phrases when necessary to maintain coverage. Phrases that contain words that are not aligned to any word in the counterpart language (**null-aligned words**; "i" in this example) are also included in phrases by connecting them to other phrases. This allows phrase-based models to generate these words in many-to-one translations, which is one of the advantages of phrase-based models mentioned above.

As a more formal definition of the phrases that we extract, if we have $\boldsymbol{e}_{i_1}^{i_2}$ containing the i_1th through i_2th words of E, and $\boldsymbol{f}_{j_1}^{j_2}$ containing the j_1th through j_2th words of F, this will be extracted as a phrases if and only if:

- There is at least one alignment link between the words in these phrases.
- The are no alignment links between words in $\boldsymbol{e}_{i_1}^{i_2}$ and $\boldsymbol{f}_1^{j_1-1}$ or $\boldsymbol{f}_{j_2+1}^{|F|}$, and similarly no alignment links between $\boldsymbol{f}_{j_1}^{j_2}$ and $\boldsymbol{e}_1^{i_1-1}$ or $\boldsymbol{e}_{i_2+1}^{|E|}$.

The first restriction is to ensure that we don't "hallucinate" phrases that contain no aligned words. The second restriction is to ensure that we don't include phrases that have only part of the necessary content. An example of this would be $\boldsymbol{f}_2^4 \leftarrow \boldsymbol{e}_3^4$ ("una manzana roja ← a red"), which violates this restriction because f_3 ("manzana"), which is included, is aligned to e_4 ("apple"), which is not included.

[38]In interest of saving memory, it is common to limit the length of phrases to some number such as 5 or 7, although there are methods around this limitation using efficient memory structures such as suffix trees [8].

A precise and efficient algorithm to extract these phrases (introduced in Figure 5.1 of [9]) is shown in Algorithm 6. Here, in Lines 2-3, we loop over all substrings in E. In Line 4, we declare a value TP ("target phrase") which contains all positions in F that correspond to this substring $\boldsymbol{e}_{i_1}^{i_2}$. In Line 5, we check if these values are quasi-consecutive, which means that all the indices in the set are consecutive, with the exception of indices that are not aligned to any word in E. In Line 6-7, we calculate the span j_1 to j_2 in F. In Line 8 we calculate all positions in E that correspond to $\boldsymbol{f}_{j_1}^{j_2}$, and in Line 9 we confirm that these indices indeed fall between i_1 and i_2. Line 10 adds this phrase to the set of extracted phrase. The loop in Lines 11-18, and inner loop in Lines 13-16 respectively expand the source phrase $\boldsymbol{f}_{j_1}^{j_2}$ on the left side and right side, adding unaligned words.

Algorithm 6 The phrase extraction algorithm. Here $A(i,j)$ indicates that e_i is aligned to f_j.

```
1: procedure PhraseExtract($E = \boldsymbol{e}_1^{|E|}, F = \boldsymbol{f}_1^{|F|}, A$)
2:     for $i_1$ from 1 to $|E|$ do
3:         for $i_2$ from $i_1$ to $|E|$ do
4:             $TP := \{j \mid \exists_i : i_1 \leq i \leq i_2 \wedge A(i,j)\}$
5:             if quasi-consecutive($TP$) then
6:                 $j_1 = \min TP$
7:                 $j_2 = \max TP$
8:                 $SP := \{i \mid \exists_j : j_1 \leq j \leq j_2 \wedge A(i,j)\}$
9:                 if $SP \subset \{i_1, i_1+1, \ldots, i_2\}$ then
10:                    $BP := BP \cup \{(\boldsymbol{e}_{i_1}^{i_2}, \boldsymbol{f}_{j_1}^{j_2}\}$
11:                    while $j_1 > 0 \wedge \forall i : A(i, j_1) = 0$ do
12:                        $j' := j_2$
13:                        while $j' \leq J \wedge \forall i : A(i, j') = 0$ do
14:                            $BP := BP \cup \{(\boldsymbol{e}_{i_1}^{i_2}, \boldsymbol{f}_{j_1}^{j'}\}$
15:                            $j' := j' + 1$
16:                        end while
17:                        $j_1 := j_1 - 1$
18:                    end while
19:                end if
20:            end if
21:        end for
22:    end forreturn $BP$
23: end procedure
```

This algorithm will extract many instances of phrases, resulting in a count $c(\boldsymbol{f}, \boldsymbol{e})$ of the number of times a particular phrase has been extracted. From these counts, we can directly calculate phrase translation probabilities $P(\boldsymbol{f} \mid \boldsymbol{e})$ that are used in Equation 133.

13.4 Phrase-based Translation with Reordering

It should be noted that the method described so far does not have any capacity for reordering of the phrases themselves. For many tasks, this reordering is not necessary, and even for tasks where reordering is essential such as MT, it is possible to handle local word reorderings to some extent by memorizing local reorderings as parts of phrases (as mentioned in Section 13.1). In language pairs (such as English-Spanish, as noted by [1]), just handling this local reordering can get us pretty far.

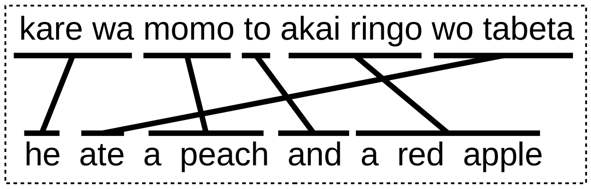
kare wa momo to akai ringo wo tabeta
he ate a peach and a red apple

Figure 43: An example of a phrase-based translation with inter-phrase reordering (for the phrase "wo tabeta/ate".

But for other language pairs with more disparate word order (e.g. German-English or Japanese-English) this is not sufficient. Thus, we would like to create a model that allows for examples that perform reordering of the phrases themselves, such as the Japanese-English example shown in Figure 43. In this example, we need to move the verb phrase "wo tabeta" at the end of the Japanese sentence to "ate" near the beginning of the English sentence.

The basic idea of how we do this reordering is by relaxing the order with which we process the source phrases. We still generate the target sentence E in order, but do not necessarily translate the phrases in F sequentially, instead translating them in arbitrary order. This indicates that, like the IBM Models, we will have to have some concept of an "alignment" between phrases indicating which phrase translates into which. Thus, we re-structure Equation 133 as follows:

$$P(\bar{F} \mid \bar{E}) = \prod_{t=1}^{|\bar{E}|} P(\bar{\boldsymbol{f}}_{\bar{a}_t}) \mid \bar{\boldsymbol{e}}_t P(\bar{a}_t \mid \bar{a}_1^{t-1}). \tag{134}$$

This indicates that we first select which phrase to translate next $\bar{a}_t$, then calculate the probability of its translation.

The probability $P(\bar{a}_t \mid \bar{a}_1^{t-1})$ is called the **reordering model**. The calculation of the reordering model is an area of active research, and there are quite a number of different methods [3, 4]. Of the various methods, the lexicalized reordering model of [11] is perhaps the most popular. The basic idea of this model is that based on the identity of the previous phrase $\langle \bar{\boldsymbol{f}}_{t-1}, \bar{\boldsymbol{e}}_{\bar{a}_{t-1}} \rangle$, we want to estimate whether the reordering satisfies one of the following patterns:

Monotonic: No reordering occurs: $\bar{a}_{t-1} + 1 = \bar{a}_t$.

Swap: A short-distance reordering that swaps the two phrases occurs: $\bar{a}_{t-1} - 1 = \bar{a}_t$.

Discontinuous: Some other variety of reordering.

These probabilities are generally estimated directly from data using maximum likelihood estimation. It should also be noted that we also need to keep track of the **coverage** of the phrases, so we will not select the same alignment twice in a single sentence, and thus $P(a_t = i) = 0$ if i has already been already used in a previous time step.

13.5 Search in Phrase-based Translation Models

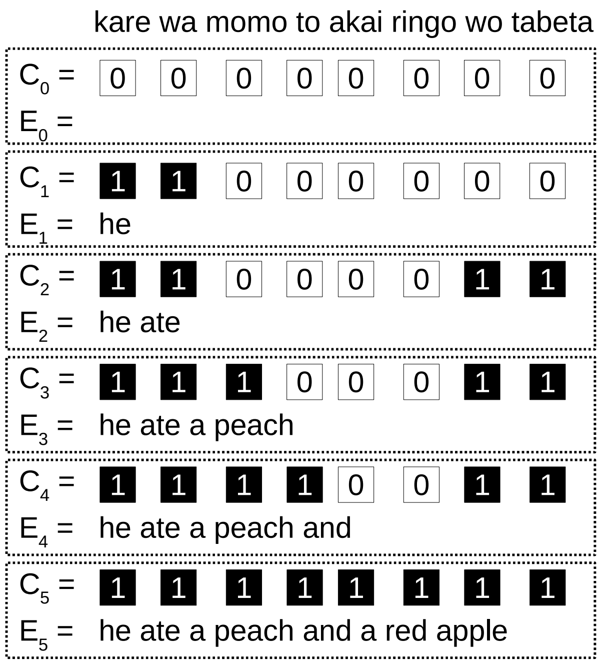

Figure 44: An example of the generation process for phrase-based MT.

Now that we have a model that can handle reordering, we need to have an algorithm to generate translations. The generation process for the previous example is shown in Figure 44. Here, at each time step we have the target sentence at that point $E_t = \text{concat}(\bar{\boldsymbol{e}}_1^t)$, and a coverage vector, which is a binary vector C_t the same length as the source sentence, indicating which source words have been covered already by a phrase that we've used so far.

This example is for a single translation E (and derivation D, expressing the phrases segmentation and alignment), but when actually generating translations it is necessary to be able to search over the space of all possible derivations. As we've seen in the previous sections, it is possible to perform search if we can create a WFST model that allows for this reordering. Unfortunately, it is not easy to create a general-purpose WFST that will do this reordering for all sentences.

What we do instead is create a per-sentence expansion of all of the phrases as a **search graph**, which represents a WFST while not being strictly similar, similar to the search graph for neural MT systems shown in Section 7. An abbreviated example of such a search graph is shown in Figure 45. Each node represents a coverage vector, and each edge between nodes represents the translation of a particular phrase. The score of these edges would be the sum of the negative log phrase translation and reordering probabilities from Equation 134. Like the word-based models in the previous section, this model can be composed with a WFST representing a language model, allowing for the full calculation of the conditional probability $P(E \mid F)$.

It should be noted that this search graph is extremely large. Even if we only consider one-word phrases, and each word only has a single translation, just considering all the permu-

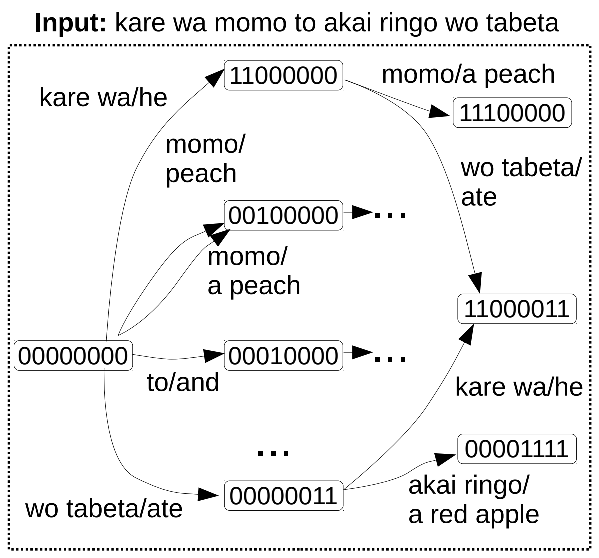

Figure 45: An example of a search graph for phrase-based translation with reordering. For brevity weights are omitted from the edges, but these weight will be equal to $-\log P(\bar{\boldsymbol{f}}|\bar{\boldsymbol{e}})$ for all phrase pairs.

tations alone will result in an exponential number of combinations. In fact, it can be shown that translation with reordering in a word-based model with a 2-gram language model is an NP-hard problem [5], and thus we cannot reasonably expect to solve it exactly.[39]

As a result, we must resort to approximate search methods, such as the beam search that we used with neural MT models in subsubsection 7.2.2. The algorithm for phrase-based MT is basically the same as that used in neural MT, we step through the translation word-by-word, expand all our hypotheses, and keep only the best k. However, there is one slight complication: in phrase-based translation, there will be some phrases that consume a single word, and some phrases that consume multiple words, which means that even after we have expanded a single phrase, some of the hypotheses will be further along in their path towards the end of the sentence in Figure 45. To ensure that hypotheses that have progressed a similar amount are compared fairly, we use a **multi-stack search algorithm**, which maintains different "stacks" based on the number of covered words in the current partial hypothesis [12], and only compares the scores of hypotheses within each of these stacks when deciding which hypothesis to expand next.

13.6 A Log-linear Formulation for Translation

Finally, we will touch briefly upon an important extension that is widely used in symbolic translation models (among many other models in NLP): **log-linear models** [10]. To grasp the basic idea behind the log-linear models, first consider that in the following formulation we have four different models that participate in our translation: the language model $P(E)$,

[39]Although with some tricks, it is possible to perform exact decoding for many, but not all sentences [2].

the phrase segmentation model $P(\bar{E} \mid E)$, the reordering model $P(A \mid \bar{E})$,[40] and the phrase translation model $P(\bar{F} \mid A, \bar{E})$. Normally these are multilpied together as

$$P(F, D, E) = P(\bar{F} \mid A, \bar{E}) P(A \mid \bar{E}) P(\bar{E} \mid E) P(E). \tag{135}$$

Equivalently, we can add together the log probabilities:

$$\log P(F, D, E) = \log P(\bar{F} \mid A, \bar{E}) + \log P(A \mid \bar{E}) + \log P(\bar{E} \mid E) + \log P(E). \tag{136}$$

The basic idea behind log-linear models is that we would like to generalize this equation by adding *weights* λ to each of these log-probabilities as follows:[41]

$$\log P(F, D, E) \propto \lambda_{\text{TM}} \log P(\bar{F} \mid A, \bar{E}) + \lambda_{\text{RM}} \log P(A \mid \bar{E}) + \lambda_{\text{SM}} \log P(\bar{E} \mid E) + \lambda_{\text{LM}} \log P(E). \tag{137}$$

This allows us to modify the translation probability, giving relatively more weight to some of the component models.

The motivation for this is two-fold. First, we would like to compensate for *imperfections in modeling.* While Equation 136 is exact, we have no guarantee that we will be able to accurately create each of these four component models and all models will all make modeling errors, some more egregious than others. By adding weights, we can decide which models we would like to trust more, and which models we would like to put less weight on, potentially increasing modeling accuracy.

The second motivation is that this formulation actually allows us to add *additional feature functions* that would not fit into the framework easily if we had to deal strictly with conditional probabilities that we could multiply together sequentially like we have been doing previously. In fact, we can now generalize our equation to the following:

$$\log P(F, D, E) \propto \sum_i \lambda_i \phi_i(F, D, E), \tag{138}$$

where $\phi_i(\cdot)$ is a feature function calculating some salient piece of information regarding the source and target sentences and the derivation. These features can, of course, include the log probabilities in Equation 137, but this framework also frees us to add additional feature functions that may be useful. For example, it is common to add:

Word Penalty: A feature $\phi_{\text{WP}}(E) = |E|$, which calculates the length of the target sentence. If $\lambda_{\text{WP}} > 0$, then the model will prefer longer sentences, and if $\lambda_{\text{WP}} < 0$ the model will prefer shorter sentences.

Direct Translation Model: A feature that calculates not $P(\bar{\boldsymbol{f}}_{a_t} \mid \bar{\boldsymbol{e}}_t)$ for every phrase, but $P(\bar{\boldsymbol{e}}_t \mid \bar{\boldsymbol{f}}_{a_t})$. Considering the probabilities in both directions helps particularly in cases where $\bar{\boldsymbol{e}}_t$ is relatively rare, and thus its conditional probability has been estimated from an insufficient amount of data.

[40] In actuality, parts of A may rely on previously selected phrases in $\bar{F}$, but for simplicity here we ignore this fact.

[41] The symbol $\propto$ means "is proportional to". In the case of log-linear models we would need to re-normalize the value to get an actual probability distribution that sums to 1. However if our only purpose is to find the translation hypothesis with the best probability, we don't need to worry about this normalization term.

Lexical Translation Model: Features, in both directions, that calculate the probability of phrases by the probabilities of their component words [6]. This can be helpful for longer phrases that themselves are rare and thus do not have well-estimated probabilities, but are composed of words whose probabilities are easier to estimate.

Now, the only question is how we calculate the weights λ. One simple way to do so is using **grid search**, trying a bunch of values in a systematic fashion and seeing which one makes it possible to achieve the best translation accuracy. Of course, there are more sophisticated methods as well, which we will cover in detail in Section 16.

13.7 Exercise

In the exercise this time we will convert our monotonic word-based translation model from the previous section to a phrase-based model. This will entail two improvements:

- Implement the phrase extraction algorithm in Algorithm 6, and run it over the word alignments obtained by your word alignment code from the exercise in Section 11.
- Implement code to convert this into a WFST.

The remainder will be very similar to what you did for the word-based models.

One possible improvement includes implementing a log-linear model, which will allow you to introduce word or phrase penalties, or weight each component model of the phrase-based translation model. Implementing reordering within the model is another interesting, but potentially challenging extension.

References

[1] Francisco Casacuberta and Enrique Vidal. Machine translation with inferred stochastic finite-state transducers. *Computational Linguistics*, 30(2), 2004.

[2] Yin-Wen Chang and Michael Collins. Exact decoding of phrase-based translation models through lagrangian relaxation. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 26–37, 2011.

[3] Michel Galley and Christopher D. Manning. A simple and effective hierarchical phrase reordering model. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 848–856, 2008.

[4] Isao Goto, Masao Utiyama, Eiichiro Sumita, Akihiro Tamura, and Sadao Kurohashi. Distortion model considering rich context for statistical machine translation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 155–165, 2013.

[5] Kevin Knight. Decoding complexity in word-replacement translation models. *Computational Linguistics*, 25(4), 1999.

[6] Philipp Koehn, Amittai Axelrod, Alexandra Birch Mayne, Chris Callison-Burch, Miles Osborne, and David Talbot. Edinburgh system description for the 2005 IWSLT speech translation evaluation. In *Proceedings of the 2005 International Workshop on Spoken Language Translation (IWSLT)*, 2005.

[7] Phillip Koehn, Franz Josef Och, and Daniel Marcu. Statistical phrase-based translation. In *Proceedings of the 2003 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics (HLT-NAACL)*, pages 48–54, 2003.

[8] Adam Lopez. Hierarchical phrase-based translation with suffix arrays. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 976–985, 2007.

[9] Franz Josef Och. *Statistical machine translation: from single-word models to alignment templates.* PhD thesis, RWTH Aachen, 2002.

[10] Franz Josef Och and Hermann Ney. Discriminative training and maximum entropy models for statistical machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 295–302, 2002.

[11] Christoph Tillman. A unigram orientation model for statistical machine translation. In *Proceedings of the 2004 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics (HLT-NAACL)*, pages 101–104, 2004.

[12] Ye-Yi Wang and Alex Waibel. Decoding algorithm in statistical machine translation. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 366–372, 1997.