21 Advanced Topics 3: Sub-word MT

Up until this point, we have treated words as the atomic unit that we are interested in training on. However, this has the problem of being less robust to low-frequency words, which is particularly a problem for neural machine translation systems that have to limit their vocabulary size for efficiency purposes. In this chapter, we first discuss a few of the phenomena that cannot be easily tackled by pure word-based approaches but can be handeled if we look at the word's characters, and then discuss some methods to handle these phenomena.

21.1 Tokenization

Before we start talking about subword-based models, it is important to consider what a word is anyway!

In English, a language where words are split by white-space, it may seem obvious: a word is something that has spaces around it. However, one obvious exception to this is punctuation: if we have the sentence "hello, friend", it would not be advantageous to treat "hello," (with a comma at the end) as a separate word from "hello" (without a comma). Thus, it is useful to perform **tokenization** before performing translation. For English, tokenization is relatively simple, and often involves splitting off punctuation, and also doing things like splitting words like "don't" into "do n't." While there are many different tokenizers, a popular ones widely used in MT is the tokenizer included in the Moses toolkit.[60].

One extreme example of the necessity for for tokenization is in languages that do not explicitly mark words with white space delimiting word boundaries. These languages include Chinese, Japanese, Thai, and several others. In these languages, it is common to create a **word segmenter** trained on data manually annotated with word boundaries, then apply this to the training and testing data for the machine translation system. In these languages, the accuracy of word segmentation has a large impact on results, with poorly segmented words often being translated incorrectly, often as unknown words. In particular, [4] note that it is extremely important to have a consistent word segmentation algorithm that usually segments words into the same units regardless of context. This is due to the fact that any differences in segmentation between the MT training data and the incoming test sentence may result in translation rules or neural net statistics not appropriately covering the mis-segmented word. As a result, it may be preferable to use a less accurate but more consistent segmentation when such a trade-off exists.

Another thing to be careful about, whether performing simple tokenization or full word segmentation, is how to handle tokenization down-stream when either performing evaluation or actually showing results to human users/evaluators. When showing results to humans, it is important to perform **detokenization**, which reverses any tokenization and outputs naturally segmented text. When evaluating results automatically, for example, using BLEU, it is important to ensure that the tokenization of the system output matches the tokenization of the reference, as described in detail by [22]. Some evaluation toolkits, such as SacreBLEU[61] or METEOR [62] take this into account automatically: they assume that you will provide them detokenized (i.e. naturally tokenized) input, and perform their own internal tokenization

[60] `http://www.statmt.org/moses/`
[61] `https://github.com/mjpost/sacreBLEU`
[62] `https://www.cs.cmu.edu/~alavie/METEOR/`

automatically at evaluation time.

21.2 Sub-word Phenomena

Now that we have discussed words, we can discuss the large number of examples in which subword structure can be useful for translation systems, a few of which are outlined in Figure 62.

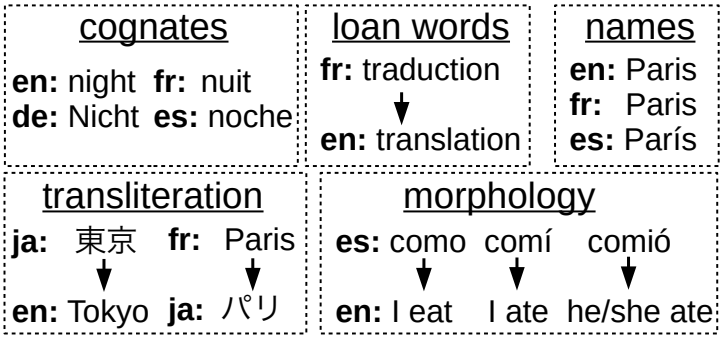

Figure 62: An example of phenomena for which sub-word information is useful.

For these words, the surface form of the words shows some non-random similarity between the source and target languages. In the extreme, we can think of examples where the words are exactly the same between the source and target sentences. For example, this is common when translating proper names, such as the "Paris" in the top-right of the figure. This can be handled by copying words directly from the source to target, as described in previous chapters (Section 20.3, [11])

However, there are many cases where words are similar, but not exactly the same. For example, this is true for **cognates**, words which share a common origin but have diverged at some point in the evolution of respective languages. For example, the word "night" in English is shared in some form with the words "Nacht" in German, "nuit" in French, and "noche" in Spanish. These reflect the fact that "night" in English descended from "nakht" in proto-Germanic (shared with German), which in turn descended from "nekwt" in proto-Indo-European (shared with all four languages above) [21, 12]. This is also true for **loan words**, which are not a result of gradual change in language, but are instead borrowed as-is from another language. One example of a loan word is "translation" (as well as most other words that end with "-ion" in English), which was borrowed from its French counterpart. While these words are not exactly the same, precluding the use of a copy mechanism, models that can appropriately handle these similarities could improve accuracy for these phenomena.

Another phenomenon that is worth noting is **transliteration**. Transliteration is the process of converting words with identical or similar pronunciations from one script to another. For example, Japanese is written in a different script than European languages, and thus words such as "Tokyo" and "Paris", which are pronounced similarly in both languages, must nevertheless be converted appropriately.

Finally, **morphology** is another notable phenomenon that affects, and requires handling of, subword structure. Morphology is the systematic changing of word forms according to their grammatical properties such as tense, case, gender, part of speech, and others. In the

example above, the Spanish verb changes according to the tense (present or past) as well as the person of the subject (first or third). These sorts of systematic changes are not captured by word-based models, but can be captured by models that are aware of some sort of subword structure.

In the following sections, we will see how to design models to handle these phenomena.

21.3 Character-based Translation

The first, and simplest, method for moving beyond words as the atomic unit for translation is to perform **character-based translation**, simply using characters to perform translation between the. In other words, instead of treating words as the symbols in F and E, we simply treat characters as the symbols in these sequences. Because neural MT methods inherently capture long-distance context through the use of recurrent neural networks or transformers, competitive results can actually be achieved without explicit segmentation into phrases [6].

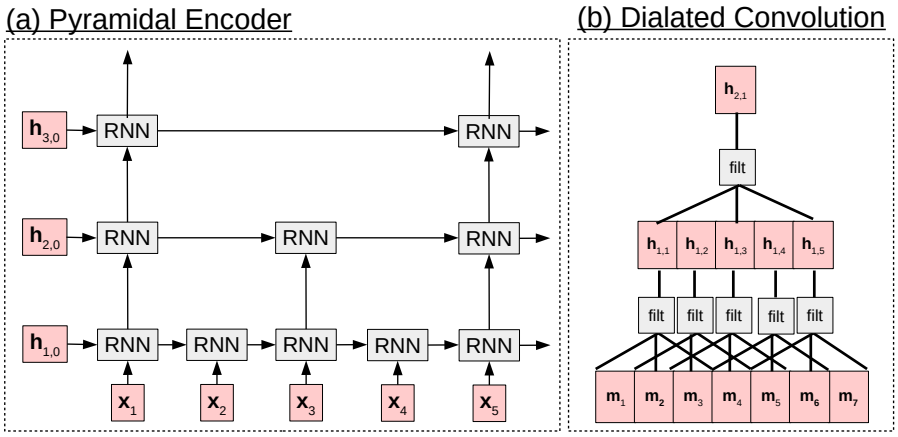

Figure 63: Encoders that reduce the resolution of input.

There are also a number of methods that attempt to create models that are character-aware, but nonetheless incorporate the idea that we would like to combine characters into units that are approximately the same size as a word. A first example is the idea of **pyramidal encoders** [3]. The idea behind this method is that we have multiple levels of stacked encoders where each successive level of encoding uses a coarser granularity. For example, the pyramidal encoder shown on the left side of Figure 63 takes in every character at its first layer, but each successive layer only takes the output of the first layer every two time steps, reducing the resolution of the output by two. A very similar idea in the context of convolutional networks is **dilated convolutions** [30], which perform convolutions that skip time steps in the middle, as shown in the right side of Figure 63.

One other important consideration for character-based models (both neural and symbolic) is their computational burden. With respect to neural models, one very obvious advantage from the computational point of view is that using characters limits the size of the output vocabulary, reducing the computational bottleneck in calculating large softmaxes over a large vocabulary of words. On the other hand, the length of the source and target sentence will be significantly longer (multiplied by the average length of a word), which means that the

number of time steps required for each sentence will increase significantly. This is an even larger problem when using a mechanism like attention, which has computation time quadratic in the length of the input. In addition, [19] report that a larger hidden layer size is necessary to efficiently capture intra-word dynamics for character-based models, resulting in an increase in a further increase in computation time.

Notably, these computation times are reduced when using pyramidal encoders or dilated convolutions, because these methods reduce the number of time steps as they progressively move through multiple layers. One issue with these models is that the reduced time steps might not correspond to linguistically meaningful units; for example "happy birthday", segmented into 4-character chunks would be "happ y_bi rthd ay" where none of these chunks are particularly linguistically meaningful. As a result, there have also been methods that attempt to jointly learn segmentation in a model similar to the pyramidal encoder, where the model is able to learn to ignore inputs from the previous layer [5].

21.4 Rethinking Tokenization: Subword Segmentation

Another way to improve translation of lower-frequency words is to come up with a new standard for tokenizing characters into words that splits words of lower frequency into smaller units. This is an extremely popular method for handling the problems mentioned above, and should likely be something you try first when you create MT or sequence-to-sequence models.

In contrast to previous methods for *supervised* tokenization segmentation where we have data manually annotated with word boundaries or human-created rules, it is also possible to perform **unsupervised word segmentation**, where original corpora (consisting of character strings) $\mathcal{F}$ and $\mathcal{E}$ are provided to a training algorithm, and boundaries splitting these into segmented corpora $\bar{\mathcal{F}}$ and $\bar{\mathcal{E}}$ are learned directly from raw text. The most prominent method for unsupervised word segmentation [10, 20] attempts to maximize the probability of this raw text using a language model:

$$\log P_{\text{LM}}(\bar{\mathcal{E}}) = \sum_{\bar{E}\in\bar{\mathcal{E}}} \sum_{t=1}^{|\bar{\mathcal{E}}|} \log P(\bar{e}_t \mid \bar{e}_1^{t-1}), \tag{213}$$

$$\text{s.t.} \forall_{\langle E,\bar{E}\rangle\in\langle\mathcal{E},\bar{\mathcal{E}}\rangle} E = \text{concat}(\bar{E}). \tag{214}$$

These models additionally add a bias against the vocabulary of the model getting too large, and describe a method to search for this maximum likelihood solution, using either an iterative procedure that re-samples the segmentations of sentences one-by-one (**Gibbs sampling**) or by heuristically prune the vocabulary by removing low-probability words until the vocabulary reaches a smaller size [15].

As a simple, faster, but potentially less accurate method for unsupervised word segmentation, [24] have recently proposed a method based on a technique called "byte pair encoding (BPE)", which is fast and relatively effective. The method finds segmentations by starting with an initial segmentation $E = \bar{E}$, where each token is its own character, then iteratively combining together the most frequent 2-gram in the corpus. The intuition behind the model is that more frequent strings (with sufficient training data) should be treated as a single unit, while less frequent strings should be split together into their component parts to prevent sparsity.

In unsupervised word segmentation algorithms, it is also common to perform **joint subword segmentation**, essentially segmenting both sides of the corpus with a single model at the same time [24]. The reason for this is because many words share character strings, as noted in Section 21.2. This would basically entail running the same unsupervised word segmentation, only over both the source and target corpora $\mathcal{F}$ and $\mathcal{E}$ at the same time, instead of running them separately. However, this method obviously will not work well for languages that do not share the same script, and also is sensitive to superficial differences in words ("revolution" in English and "revolucion" in Spanish) that result in different segmentations for very similar words.

Luckily, there are high-quality implementations of these subword segmentation algorithms that can be used out-of-the-box. One example of this is `sentencepiece`.[63].

21.5 Hybrid Word-character Models

While subword segmentation is convenient, it is also overly simple and can be sub-optimal for translation of morphologically complex languages, or for multilingual translation [28]. It is also possible to create models that work on the word level but nonetheless have a concept of the structure of words.

One example of this is models for transliteration, where the model decides to translate character-by-character only if it decides that the word should be transliterated. For example, [13] come up with a model that identifies **named entities** (e.g. people or places) in the source text using a standard named entity recognizer, and decides whether and how to transliterate it. This is a difficult problem because some entities may require a mix of translation and transliteration; for example "Carnegie Mellon University" is a named entity, but while "Carnegie" and "Mellon" may be transliterated, "University" will often be translated into the appropriate target language. [8] adapt this to be integrated in a phrase-based translation system.

Word-character hybrid models have also been implemented within the neural MT paradigm. One method, proposed by [17], uses word boundaries to specify the granularity of the encoding, so that each word is encoded by a single vector (c.f. the fixed-length encoding of the previous section). The encoding of the word can be performed using a bi-directional LSTM over its characters, with the final states in each direction being concatenated into the word representation. On the decoding side, an RNN generates word representations one-by-one, then the over-arching word representation is used to generate the target word one character at a time.

Another method for representing words effectively is based on character n-grams [23, 29]. The way this model works is that each n-character sequence is given a unique embedding, and the representation of the word is calculated as the sum of the embeddings. For example, if we were using n-grams of length 2 and 3, and the word to encode was "dogs", then the embedding for dog would be calculated as the sum of the embeddings for "_d", "do", "og", "gs", "s_", "_do", "dog", "ogs", "gs_", where the underbar represents the beginning and end of the word. This representation is relatively efficient to compute, and has proven empirically effective both in word embedding [29], and sequence-to-sequence tasks [1]. In the context of multilingual translation, this can be taken a step further, encoding words based on characters

[63] `https://github.com/google/sentencepiece`

to encourage sharing between them, but then performing language-specific transformations to account for differences in spelling [28].

Alternatively, it is also possible to use this character-based model only for words that do not exist in the vocabulary [18]. This is beneficial, as it allows the model to directly generate words that are in its vocabulary, presumably more frequent and well-learned words, while falling back to a character-based representation when it is necessary.

21.6 Models of Morphology

As mentioned above, morphology is the phenomenon of word forms being transformed in a consistent way corresponding to their syntactic function. Because these transformations are consistent across words, models that can capture these transformations appropriately can be used to more accurately translate from or to inflected word forms. One common way to incorporate morphological information into MT is through a three step process of (1) *analysis*, which calculates various features of the word to be translated such as the lemma or morphological tags, (2) *translation*, which translates this factored representation making various independence assumptions, and (3) *generation*, which creates the surface form from each of these factors [14].

For the analysis and generation steps, the classical way of creating an analyzer is through manually created rules with expressed as finite-state automata that take in the characters of the word one at a time and output possible analyses [2]. This allows us to create a concise list of analysis candidates for words in the dictionary with relatively high precision, and these methods are still widely used for a number of languages for which good morphological analyzers exist. However, these methods rely on a linguist capable of making rules, an extensive dictionary in the language, and lack the ability to disambiguate hypotheses based on context. As a result, there are also methods based on symbolic [9] or neural [25] approaches to perform morphological analysis and contextual disambiguation.

Once we have an analysis of one or both sides of the translation pair, we can proceed to do translation. One easy and effective way to do so when using a morphologically rich language on the source side is simply to analyze the source corpus, split the words into their lemmas and morphological tags, and use this split data as input to a normal MT system [16]. This is very similar to the subword splitting approach in Section 21.4, so it is worth considering the differences. In the case of **concatenative morphology**, where a word can be viewed as the concatenation of its morphemes (e.g. the word "undecided" can be viewed as the concatenation of "un+decide+d"), subword splitting methods may be sufficient. However, there are also more difficult cases such as **infix morphology**, where the inner parts of a word are changed due to morphological processes (e.g. in English "goose" and its plural "geese"). In these more complicated cases, normalizing to the lemma can be an effective way to increase the generalization capabilities of the translation model.

On the other hand, if we have rich morphology on the target side, we can also think of converting our target corpus into a sequence of lemmas and morphological tags and translating into these [27, 7]. Before showing the result to the end user, we need to generate the surface form from these lemmas and tags (e.g. going from "goose +PLURAL" to "geese"). Similarly, this process can be done with a rule-based generation model created by a linguist, or through data-driven approaches, much like morphological analysis. One thing to note is that in general, translation into morphologically rich languages is considered more difficult than translation

from morphologically rich languages to a morphologically poor language such as English. The reason for this is twofold. First, morphologically rich languages tend to require more long-distance agreement between the forms of words; similarly to how the subject being first, second, or third person affects the conjugation of the verb (e.g. "I run" vs. "he runs"), in morphologically rich languages it is not uncommon for the person, case, gender, or other information to be matched between different words in the sentence. Second, by adding an extra generation step, it is common for error propagation to occur, with errors in the first step cascading to errors in the second step.

21.7 Further Reading

There are a number of additional topics related to sub-word models that interested readers can examine:

Factored translation models: Factored translation models split a word up into several factors and translate them given some independence assumptions on what factors influence others [14]. For example, a word may be split into a lemma factor, a tense factor, and a plural factor, each of which would be translated independently, then combined together in a final generation step. This is different from the simple pre/post-processing approaches described above in that these various factors are tightly integrated within the translation model, and translation is still performed on a word-by-word basis.

Considering multiple segmentations: One problem with selecting segmentations at is that there may be multiple ambiguous ways to segment a particular word, and it is not necessarily the case that we can determine which one is best. [15] propose a method to fix these problems at training time by randomly sampling which segmentation to use according to a probabilistic model. Further, [26] propose methods to fix this problem at test time by translating from a lattice of segmentation candidates.

21.8 Exercise

As an exercise, you could try to either

1. Train a character-based neural machine translation system using your existing code. Note its advantages and disadvantages compared to your existing word-based model, including training speed and accuracy.

2. Train a system using byte-pair encoding. This would entail implementing the byte pair encoding algorithm, and using it as a pre-processing step before running your normal system.

References

[1] Duygu Ataman and Marcello Federico. Compositional representation of morphologically-rich input for neural machine translation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 305–311. Association for Computational Linguistics, 2018.

[2] Kenneth R. Beesley and Lauri Karttunen. *Finite State Morphology*. CSLI Studies in Computational Linguistics. CSLI Publications, 2003.

[3] William Chan, Navdeep Jaitly, Quoc Le, and Oriol Vinyals. Listen, attend and spell: A neural network for large vocabulary conversational speech recognition. In *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 4960–4964. IEEE, 2016.

[4] Pi-Chuan Chang, Michel Galley, and Christopher D. Manning. Optimizing Chinese word segmentation for machine translation performance. In *Proceedings of the 3rd Workshop on Statistical Machine Translation (WMT)*, pages 224–232, 2008.

[5] Colin Cherry, George Foster, Ankur Bapna, Orhan Firat, and Wolfgang Macherey. Revisiting character-based neural machine translation with capacity and compression. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2018.

[6] Junyoung Chung, Kyunghyun Cho, and Yoshua Bengio. A character-level decoder without explicit segmentation for neural machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1693–1703, 2016.

[7] Ann Clifton and Anoop Sarkar. Combining morpheme-based machine translation with post-processing morpheme prediction. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics (ACL)*, 2011.

[8] Nadir Durrani, Hassan Sajjad, Hieu Hoang, and Philipp Koehn. Integrating an unsupervised transliteration model into statistical machine translation. In *Proceedings of the 14th European Chapter of the Association for Computational Linguistics (EACL)*, pages 148–153, 2014.

[9] Greg Durrett and John DeNero. Supervised learning of complete morphological paradigms. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 1185–1195, 2013.

[10] Sharon Goldwater, Thomas L. Griffiths, and Mark Johnson. A Bayesian framework for word segmentation: Exploring the effects of context. *Cognition*, 112(1), 2009.

[11] Jiatao Gu, Zhengdong Lu, Hang Li, and Victor O.K. Li. Incorporating copying mechanism in sequence-to-sequence learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1631–1640, 2016.

[12] Douglas Harper et al. Online etymology dictionary, 2001.

[13] Kevin Knight and Jonathan Graehl. Machine transliteration. *Computational Linguistics (CL)*, 24(4):599–612, 1998.

[14] Philipp Koehn and Hieu Hoang. Factored translation models. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, 2007.

[15] Taku Kudo. Subword regularization: Improving neural network translation models with multiple subword candidates. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 66–75, 2018.

[16] Young-Suk Lee. Morphological analysis 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 57–60, 2004.

[17] Wang Ling, Isabel Trancoso, Chris Dyer, and Alan W Black. Character-based neural machine translation. *arXiv preprint arXiv:1511.04586*, 2015.

[18] Minh-Thang Luong and Christopher D. Manning. Achieving open vocabulary neural machine translation with hybrid word-character models. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1054–1063, Berlin, Germany, August 2016. Association for Computational Linguistics.

[19] Tomáš Mikolov, Ilya Sutskever, Anoop Deoras, Hai-Son Le, and Stefan Kombrink. Subword language modeling with neural networks.

[20] Daichi Mochihashi, Takeshi Yamada, and Naonori Ueda. Bayesian unsupervised word segmentation with nested Pitman-Yor modeling. In *Proceedings of the 47th Annual Meeting of the Association for Computational Linguistics (ACL)*, 2009.

[21] Oxford English Oxford. *Oxford English Dictionary*. Oxford: Oxford University Press, 2009.

[22] Matt Post. A call for clarity in reporting BLEU scores. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 186–191, Belgium, Brussels, October 2018. Association for Computational Linguistics.

[23] Hinrich Sch utze. Word space. 5:895–902, 1993.

[24] Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1715–1725, 2016.

[25] Qinlan Shen, Daniel Clothiaux, Emily Tagtow, Patrick Littell, and Chris Dyer. The role of context in neural morphological disambiguation. In *Proceedings of the 26th International Conference on Computational Linguistics (COLING)*, pages 181–191, 2016.

[26] Jinsong Su, Zhixing Tan, Deyi Xiong, Rongrong Ji, Xiaodong Shi, and Yang Liu. Lattice-based recurrent neural network encoders for neural machine translation. pages 3302–3308, 2017.

[27] Kristina Toutanova, Hisami Suzuki, and Achim Ruopp. Applying morphology generation models to machine translation. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 514–522, 2008.

[28] Xinyi Wang, Hieu Pham, Philip Arthur, and Graham Neubig. Multilingual neural machine translation with soft decoupled encoding. In *International Conference on Learning Representations (ICLR)*, New Orleans, LA, USA, May 2019.

[29] John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu. Charagram: Embedding words and sentences via character n-grams. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1504–1515, 2016.

[30] Fisher Yu and Vladlen Koltun. Multi-scale context aggregation by dilated convolutions. 2016.