CHAPTER

13 Machine Translation

> "I want to talk the dialect of your people. It's no use of talking unless people understand what you say."
>
> Zora Neale Hurston, *Moses, Man of the Mountain* 1939, p. 121

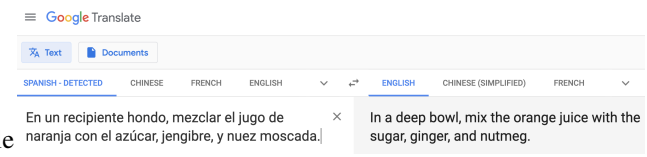
machine translation
MT

This chapter introduces **machine translation** (**MT**), the use of computers to translate from one language to another.

Of course translation, in its full generality, such as the translation of literature, or poetry, is a difficult, fascinating, and intensely human endeavor, as rich as any other area of human creativity.

information access

Machine translation in its present form therefore focuses on a number of very practical tasks. Perhaps the most common current use of machine translation is for **information access**. We might want to translate some instructions on the web, perhaps the recipe for a favorite dish, or the steps for putting together some furniture. Or we might want to read an article in a newspaper, or get information from an online resource like Wikipedia or a government webpage in some other language. MT for information access is probably one of the most common uses of NLP technology, and Google Translate alone (shown above) translates hundreds of billions of words a day between over 100 languages. Improvements in machine translation can thus help reduce what is often called the **digital divide** in information access: the fact that much more information is available in English and other languages spoken in wealthy countries. Web searches in English return much more information than searches in other languages, and online resources like Wikipedia are much larger in English and other higher-resourced languages. High-quality translation can help provide information to speakers of lower-resourced languages.



digital divide

post-editing
CAT
localization

Another common use of machine translation is to aid human translators. MT systems are routinely used to produce a draft translation that is fixed up in a **post-editing** phase by a human translator. This task is often called **computer-aided translation** or **CAT**. CAT is commonly used as part of **localization**: the task of adapting content or a product to a particular language community.

Finally, a more recent application of MT is to in-the-moment human communication needs. This includes incremental translation, translating speech on-the-fly before the entire sentence is complete, as is commonly used in simultaneous interpretation. Image-centric translation can be used for example to use OCR of the text on a phone camera image as input to an MT system to translate menus or street signs.

encoder-decoder

The standard algorithm for MT is the **encoder-decoder** network, an architecture that we introduced in Chapter 9 for RNNs. Recall that encoder-decoder or sequence-to-sequence models are used for tasks in which we need to map an input sequence to an output sequence that is a complex function of the entire input sequence. Indeed,

in machine translation, the words of the target language don't necessarily agree with the words of the source language in number or order. Consider translating the following made-up English sentence into Japanese.

(13.1) English: *He wrote a letter to a friend*
Japanese: *tomodachi ni tegami-o kaita*
friend to letter wrote

Note that the elements of the sentences are in very different places in the different languages. In English, the verb is in the middle of the sentence, while in Japanese, the verb *kaita* comes at the end. The Japanese sentence doesn't require the pronoun *he*, while English does.

Such differences between languages can be quite complex. In the following actual sentence from the United Nations, notice the many changes between the Chinese sentence (we've given in red a word-by-word gloss of the Chinese characters) and its English equivalent.

(13.2) 大会/General Assembly 在/on 1982年/1982 12月/December 10日/10 通过了/adopted 第37号/37th 决议/resolution， 核准了/approved 第二次/second 探索/exploration 及/and 和平peaceful 利用/using 外层空间/outer space 会议/conference 的/of 各项/various 建议/suggestions 。

On 10 December 1982 , the General Assembly adopted resolution 37 in which it endorsed the recommendations of the Second United Nations Conference on the Exploration and Peaceful Uses of Outer Space .

Note the many ways the English and Chinese differ. For example the ordering differs in major ways; the Chinese order of the noun phrase is "peaceful using outer space conference of suggestions" while the English has "suggestions of the ... conference on peaceful use of outer space"). And the order differs in minor ways (the date is ordered differently). English requires *the* in many places that Chinese doesn't, and adds some details (like "in which" and "it") that aren't necessary in Chinese. Chinese doesn't grammatically mark plurality on nouns (unlike English, which has the "-s" in "recommendations"), and so the Chinese must use the modifier 各项/*various* to make it clear that there is not just one recommendation. English capitalizes some words but not others. Encoder-decoder networks are very successful at handling these sorts of complicated cases of sequence mappings.

We'll begin in the next section by considering the linguistic background about how languages vary, and the implications this variance has for the task of MT. Then we'll sketch out the standard algorithm, give details about things like input tokenization and creating training corpora of parallel sentences, give some more low-level details about the encoder-decoder network, and finally discuss how MT is evaluated, introducing the simple chrF metric.

13.1 Language Divergences and Typology

universal

There are about 7,000 languages in the world. Some aspects of human language seem to be **universal**, holding true for every one of these languages, or are statistical universals, holding true for most of these languages. Many universals arise from the functional role of language as a communicative system by humans. Every language, for example, seems to have words for referring to people, for talking about eating and drinking, for being polite or not. There are also structural linguistic universals; for

example, every language seems to have nouns and verbs (Chapter 8), has ways to ask questions, or issue commands, has linguistic mechanisms for indicating agreement or disagreement.

translation divergence

Yet languages also **differ** in many ways (as has been pointed out since ancient times; see Fig. 13.1). Understanding what causes such **translation divergences** (Dorr, 1994) can help us build better MT models. We often distinguish the **idiosyncratic** and lexical differences that must be dealt with one by one (the word for "dog" differs wildly from language to language), from **systematic** differences that we can model in a general way (many languages put the verb before the grammatical object; others put the verb after the grammatical object). The study of these systematic cross-linguistic similarities and differences is called **linguistic typology**. This section sketches some typological facts that impact machine translation; the interested reader should also look into WALS, the World Atlas of Language Structures, which gives many typological facts about languages (Dryer and Haspelmath, 2013).

typology

Figure 13.1 The Tower of Babel, Pieter Bruegel 1563. Wikimedia Commons, from the Kunsthistorisches Museum, Vienna.

13.1.1 Word Order Typology

As we hinted it in our example above comparing English and Japanese, languages differ in the basic word order of verbs, subjects, and objects in simple declarative clauses. German, French, English, and Mandarin, for example, are all **SVO** (**Subject-Verb-Object**) languages, meaning that the verb tends to come between the subject and object. Hindi and Japanese, by contrast, are **SOV** languages, meaning that the verb tends to come at the end of basic clauses, and Irish and Arabic are **VSO** languages. Two languages that share their basic word order type often have other similarities. For example, **VO** languages generally have **prepositions**, whereas **OV** languages generally have **postpositions**.

SVO

SOV

VSO

Let's look in more detail at the example we saw above. In this SVO English sentence, the verb *wrote* is followed by its object *a letter* and the prepositional phrase *to a friend*, in which the preposition *to* is followed by its argument *a friend*. Arabic, with a VSO order, also has the verb before the object and prepositions. By contrast, in the Japanese example that follows, each of these orderings is reversed; the verb is *preceded* by its arguments, and the postposition follows its argument.

(13.3) English: *He wrote a letter to a friend*
Japanese: *tomodachi ni tegami-o kaita*
friend to letter wrote
Arabic: *katabt risāla li ṣadq*
wrote letter to friend

Other kinds of ordering preferences vary idiosyncratically from language to language. In some SVO languages (like English and Mandarin) adjectives tend to appear before verbs, while in others languages like Spanish and Modern Hebrew, adjectives appear after the noun:

(13.4) **Spanish** *bruja verde* **English** *green witch*

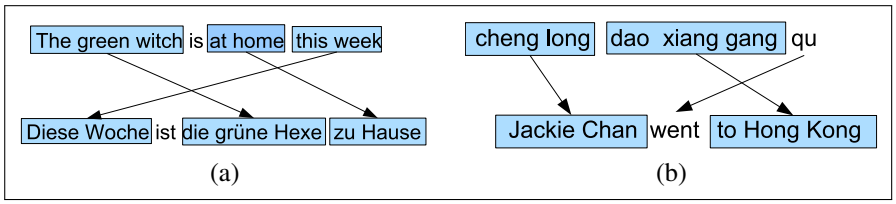


Figure 13.2 Examples of other word order differences: (a) In German, adverbs occur in initial position that in English are more natural later, and tensed verbs occur in second position. (b) In Mandarin, preposition phrases expressing goals often occur pre-verbally, unlike in English.

Fig. 13.2 shows examples of other word order differences. All of these word order differences between languages can cause problems for translation, requiring the system to do huge structural reorderings as it generates the output.

13.1.2 Lexical Divergences

Of course we also need to translate the individual words from one language to another. For any translation, the appropriate word can vary depending on the context. The English source-language word *bass*, for example, can appear in Spanish as the fish *lubina* or the musical instrument *bajo*. German uses two distinct words for what in English would be called a *wall*: *Wand* for walls inside a building, and *Mauer* for walls outside a building. Where English uses the word *brother* for any male sibling, Chinese and many other languages have distinct words for *older brother* and *younger brother* (Mandarin *gege* and *didi*, respectively). In all these cases, translating *bass*, *wall*, or *brother* from English would require a kind of specialization, disambiguating the different uses of a word. For this reason the fields of MT and Word Sense Disambiguation (Chapter 23) are closely linked.

Sometimes one language places more grammatical constraints on word choice than another. We saw above that English marks nouns for whether they are singular or plural. Mandarin doesn't. Or French and Spanish, for example, mark grammatical gender on adjectives, so an English translation into French requires specifying adjective gender.

The way that languages differ in lexically dividing up conceptual space may be more complex than this one-to-many translation problem, leading to many-to-many mappings. For example, Fig. 13.3 summarizes some of the complexities discussed by Hutchins and Somers (1992) in translating English *leg, foot*, and *paw*, to French. For example, when *leg* is used about an animal it's translated as French *jambe*; but about the leg of a journey, as French *etape*; if the leg is of a chair, we use French *pied*.

lexical gap

Further, one language may have a **lexical gap**, where no word or phrase, short of an explanatory footnote, can express the exact meaning of a word in the other language. For example, English does not have a word that corresponds neatly to Mandarin *xiào* or Japanese *oyakōkōo* (in English one has to make do with awkward phrases like *filial piety* or *loving child*, or *good son/daughter* for both).

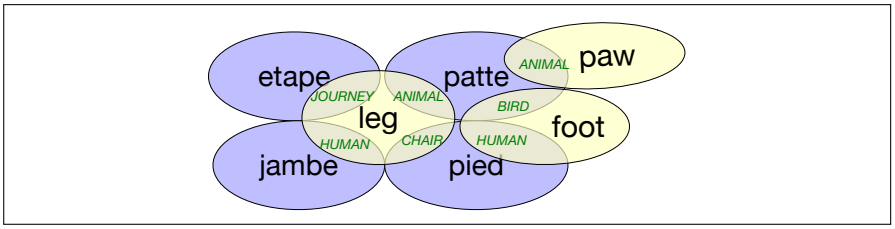


Figure 13.3 The complex overlap between English *leg*, *foot*, etc., and various French translations as discussed by Hutchins and Somers (1992).

Finally, languages differ systematically in how the conceptual properties of an event are mapped onto specific words. Talmy (1985, 1991) noted that languages can be characterized by whether direction of motion and manner of motion are marked on the verb or on the "satellites": particles, prepositional phrases, or adverbial phrases. For example, a bottle floating out of a cave would be described in English with the direction marked on the particle *out*, while in Spanish the direction would be marked on the verb:

(13.5) English: *The bottle floated out.*
Spanish: La botella salió flotando.
The bottle exited floating.

verb-framed

Verb-framed languages mark the direction of motion on the verb (leaving the satellites to mark the manner of motion), like Spanish *acercarse* 'approach', *alcanzar* 'reach', *entrar* 'enter', *salir* 'exit'. **Satellite-framed** languages mark the direction of motion on the satellite (leaving the verb to mark the manner of motion), like English *crawl out*, *float off*, *jump down*, *run after*. Languages like Japanese, Tamil, and the many languages in the Romance, Semitic, and Mayan languages families, are verb-framed; Chinese as well as non-Romance Indo-European languages like English, Swedish, Russian, Hindi, and Farsi are satellite framed (Talmy 1991, Slobin 1996).

satellite-framed

13.1.3 Morphological Typology

isolating

Morphologically, languages are often characterized along two dimensions of variation. The first is the number of morphemes per word, ranging from **isolating** languages like Vietnamese and Cantonese, in which each word generally has one morpheme, to **polysynthetic** languages like Siberian Yupik ("Eskimo"), in which a single word may have very many morphemes, corresponding to a whole sentence in

polysynthetic

English. The second dimension is the degree to which morphemes are segmentable, ranging from **agglutinative** languages like Turkish, in which morphemes have relatively clean boundaries, to **fusion** languages like Russian, in which a single affix may conflate multiple morphemes, like *-om* in the word *stolom* (table-SG-INSTR-DECL1), which fuses the distinct morphological categories instrumental, singular, and first declension.

agglutinative
fusion

Translating between languages with rich morphology requires dealing with structure below the word level, and for this reason modern systems generally use subword models like the wordpiece or BPE models of Section 13.2.1.

13.1.4 Referential density

Finally, languages vary along a typological dimension related to the things they tend to omit. Some languages, like English, require that we use an explicit pronoun when talking about a referent that is given in the discourse. In other languages, however, we can sometimes omit pronouns altogether, as the following example from Spanish shows[1]:

(13.6) [El jefe]$_i$ dio con un libro. $\emptyset_i$ Mostró su hallazgo a un descifrador ambulante.
[The boss] came upon a book. [He] showed his find to a wandering decoder.

pro-drop

Languages that can omit pronouns are called **pro-drop** languages. Even among the pro-drop languages, there are marked differences in frequencies of omission. Japanese and Chinese, for example, tend to omit far more than does Spanish. This dimension of variation across languages is called the dimension of **referential density**. We say that languages that tend to use more pronouns are more **referentially dense** than those that use more zeros. Referentially sparse languages, like Chinese or Japanese, that require the hearer to do more inferential work to recover antecedents are also called **cold** languages. Languages that are more explicit and make it easier for the hearer are called **hot** languages. The terms *hot* and *cold* are borrowed from Marshall McLuhan's 1964 distinction between hot media like movies, which fill in many details for the viewer, versus cold media like comics, which require the reader to do more inferential work to fill out the representation (Bickel, 2003).

referential density
cold language
hot language

Translating from languages with extensive pro-drop, like Chinese or Japanese, to non-pro-drop languages like English can be difficult since the model must somehow identify each zero and recover who or what is being talked about in order to insert the proper pronoun.

13.2 Machine Translation using Encoder-Decoder

The standard architecture for MT is the **encoder-decoder transformer** or **sequence-to-sequence** model, an architecture we saw for RNNs in Chapter 9. We'll see the details of how to apply this architecture to transformers in Section 13.3, but first let's talk about the overall task.

Most machine translation tasks make the simplification that we can translate each sentence independently, so we'll just consider individual sentences for now. Given a sentence in a **source** language, the MT task is then to generate a corresponding sentence in a **target** language. For example, an MT system is given an English sentence like

[1] Here we use the $\emptyset$-notation; we'll introduce this and discuss this issue further in Chapter 26

The green witch arrived

and must translate it into the Spanish sentence:

Llegó la bruja verde

MT uses supervised machine learning: at training time the system is given a large set of **parallel** sentences (each sentence in a source language matched with a sentence in the target language), and learns to map source sentences into target sentences. In practice, rather than using words (as in the example above), we split the sentences into a sequence of subword tokens (tokens can be words, or subwords, or individual characters). The systems are then trained to maximize the probability of the sequence of tokens in the target language $y_1,...,y_m$ given the sequence of tokens in the source language $x_1,...,x_n$:

$$P(y_1,\ldots,y_m|x_1,\ldots,x_n) \tag{13.7}$$

Rather than use the input tokens directly, the encoder-decoder architecture consists of two components, an **encoder** and a **decoder**. The encoder takes the input words $x = [x_1,\ldots,x_n]$ and produces an intermediate context $\mathbf{h}$. At decoding time, the system takes h and, word by word, generates the output y:

$$\mathbf{h} = \texttt{encoder}(x) \tag{13.8}$$

$$y_{i+1} = \texttt{decoder}(\mathbf{h}, y_1,\ldots,y_i)) \quad \forall i \in [1,\ldots,m] \tag{13.9}$$

In the next two sections we'll talk about subword tokenization, and then how to get parallel corpora for training, and then we'll introduce the details of the encoder-decoder architecture.

13.2.1 Tokenization

Machine translation systems use a vocabulary that is fixed in advance, and rather than using space-separated words, this vocabulary is generated with subword tokenization algorithms, like the **BPE** algorithm sketched in Chapter 2. A shared vocabulary is used for the source and target languages, which makes it easy to copy tokens (like names) from source to target. Using subword tokenization with tokens shared between languages makes it natural to translate between languages like English or Hindi that use spaces to separate words, and languages like Chinese or Thai that don't.

We build the vocabulary by running a subword tokenization algorithm on a corpus that contains both source and target language data.

Rather than the simple BPE algorithm from Fig. 2.13, modern systems often use more powerful tokenization algorithms. Some systems (like BERT) use a variant of BPE called the **wordpiece** algorithm, which instead of choosing the most frequent set of tokens to merge, chooses merges based on which one most increases the language model probability of the tokenization. Wordpieces use a special symbol at the beginning of each token; here's a resulting tokenization from the Google MT system (Wu et al., 2016):

wordpiece

words:	Jet makers feud over seat width with big orders at stake
wordpieces:	_J et _makers _fe ud _over _seat _width _with _big _orders _at _stake

The wordpiece algorithm is given a training corpus and a desired vocabulary size V, and proceeds as follows:

1. Initialize the wordpiece lexicon with characters (for example a subset of Unicode characters, collapsing all the remaining characters to a special unknown character token).
2. Repeat until there are V wordpieces:
   (a) Train an n-gram language model on the training corpus, using the current set of wordpieces.
   (b) Consider the set of possible new wordpieces made by concatenating two wordpieces from the current lexicon. Choose the one new wordpiece that most increases the language model probability of the training corpus.

Recall that with BPE we had to specify the number of merges to perform; in wordpiece, by contrast, we specify the total vocabulary, which is a more intuitive parameter. A vocabulary of 8K to 32K word pieces is commonly used.

unigram
SentencePiece

An even more commonly used tokenization algorithm is (somewhat ambiguously) called the **unigram** algorithm (Kudo, 2018) or sometimes the **SentencePiece** algorithm, and is used in systems like ALBERT (Lan et al., 2020) and T5 (Raffel et al., 2020). (Because unigram is the default tokenization algorithm used in a library called SentencePiece that adds a useful wrapper around tokenization algorithms (Kudo and Richardson, 2018b), authors often say they are using SentencePiece tokenization but really mean they are using the **unigram** algorithm)).

In unigram tokenization, instead of building up a vocabulary by merging tokens, we start with a huge vocabulary of every individual unicode character plus all frequent sequences of characters (including all space-separated words, for languages with spaces), and iteratively remove some tokens to get to a desired final vocabulary size. The algorithm is complex (involving suffix-trees for efficiently storing many tokens, and the EM algorithm for iteratively assigning probabilities to tokens), so we don't give it here, but see Kudo (2018) and Kudo and Richardson (2018b). Roughly speaking the algorithm proceeds iteratively by estimating the probability of each token, tokenizing the input data using various tokenizations, then removing a percentage of tokens that don't occur in high-probability tokenization, and then iterates until the vocabulary has been reduced down to the desired number of tokens.

Why does unigram tokenization work better than BPE? BPE tends to creates lots of very small non-meaningful tokens (because BPE can only create larger words or morphemes by merging characters one at a time), and it also tends to merge very common tokens, like the suffix *ed*, onto their neighbors. We can see from these examples from Bostrom and Durrett (2020) that unigram tends to produce tokens that are more semantically meaningful:

Original:	`corrupted`	Original:	`Completely preposterous suggestions`
BPE:	`cor rupted`	BPE:	`Comple t ely prep ost erous suggest ions`
Unigram:	`corrupt ed`	Unigram:	`Complete ly pre post er ous suggestion s`

13.2.2 Creating the Training data

parallel corpus
Europarl

Machine translation models are trained on a **parallel corpus**, sometimes called a **bitext**, a text that appears in two (or more) languages. Large numbers of parallel corpora are available. Some are governmental; the **Europarl** corpus (Koehn, 2005), extracted from the proceedings of the European Parliament, contains between 400,000 and 2 million sentences each from 21 European languages. The United Nations Parallel Corpus contains on the order of 10 million sentences in the six official languages of the United Nations (Arabic, Chinese, English, French, Russian, Spanish) Ziemski et al. (2016). Other parallel corpora have been made from movie and

TV subtitles, like the **OpenSubtitles** corpus (Lison and Tiedemann, 2016), or from general web text, like the **ParaCrawl** corpus of 223 million sentence pairs between 23 EU languages and English extracted from the CommonCrawl Bañón et al. (2020).

Sentence alignment

Standard training corpora for MT come as aligned pairs of sentences. When creating new corpora, for example for underresourced languages or new domains, these sentence alignments must be created. Fig. 13.4 gives a sample hypothetical sentence alignment.

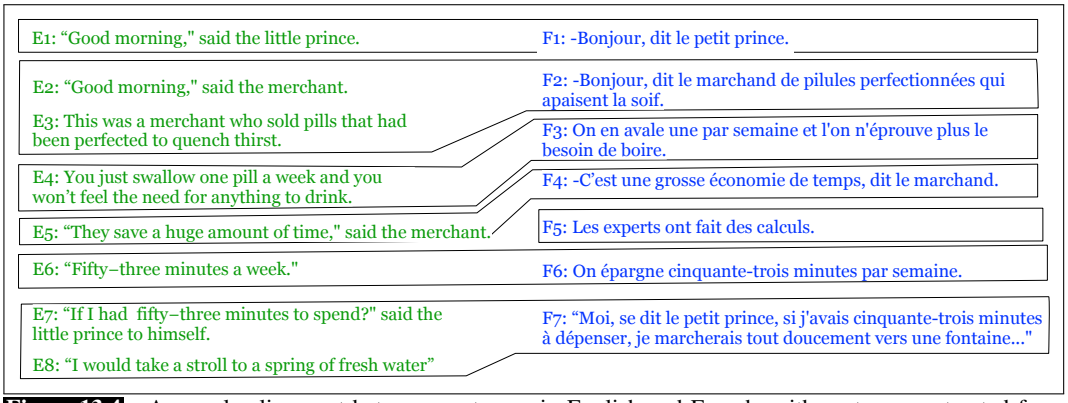


Figure 13.4 A sample alignment between sentences in English and French, with sentences extracted from Antoine de Saint-Exupery's *Le Petit Prince* and a hypothetical translation. Sentence alignment takes sentences $e_1,...,e_n$, and $f_1,...,f_n$ and finds minimal sets of sentences that are translations of each other, including single sentence mappings like (e_1,f_1), (e_4,f_3), (e_5,f_4), (e_6,f_6) as well as 2-1 alignments (e_2/e_3,f_2), (e_7/e_8,f_7), and null alignments (f_5).

Given two documents that are translations of each other, we generally need two steps to produce sentence alignments:

- a cost function that takes a span of source sentences and a span of target sentences and returns a score measuring how likely these spans are to be translations.
- an alignment algorithm that takes these scores to find a good alignment between the documents.

To score the similarity of sentences across languages, we need to make use of a **multilingual embedding space**, in which sentences from different languages are in the same embedding space (Artetxe and Schwenk, 2019). Given such a space, cosine similarity of such embeddings provides a natural scoring function (Schwenk, 2018). Thompson and Koehn (2019) give the following cost function between two sentences or spans x,y from the source and target documents respectively:

$$c(x,y) = \frac{(1-\cos(x,y))\text{nSents}(x)\ \text{nSents}(y)}{\sum_{s=1}^{S} 1-\cos(x,y_s) + \sum_{s=1}^{S} 1-\cos(x_s,y)} \tag{13.10}$$

where nSents() gives the number of sentences (this biases the metric toward many alignments of single sentences instead of aligning very large spans). The denominator helps to normalize the similarities, and so $x_1,...,x_S,y_1,...,y_S$, are randomly selected sentences sampled from the respective documents.

Usually dynamic programming is used as the alignment algorithm (Gale and Church, 1993), in a simple extension of the minimum edit distance algorithm we introduced in Chapter 2.

Finally, it's helpful to do some corpus cleanup by removing noisy sentence pairs. This can involve handwritten rules to remove low-precision pairs (for example removing sentences that are too long, too short, have different URLs, or even pairs that are too similar, suggesting that they were copies rather than translations). Or pairs can be ranked by their multilingual embedding cosine score and low-scoring pairs discarded.

13.3 Details of the Encoder-Decoder Model

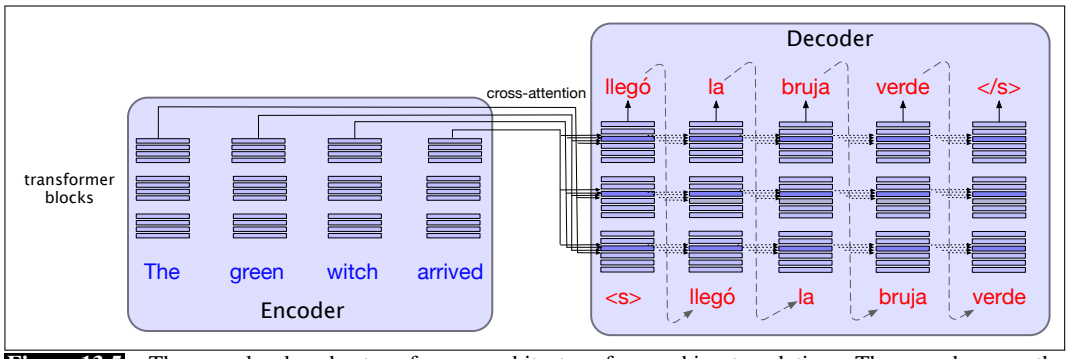


Figure 13.5 The encoder-decoder transformer architecture for machine translation. The encoder uses the transformer blocks we saw in Chapter 9, while the decoder uses a more powerful block with an extra **cross-attention** layer that can attend to all the encoder words. We'll see this in more detail in the next section.

The standard architecture for MT is the encoder-decoder transformer. The encoder-decoder architecture was introduced already for RNNs in Chapter 9, and the transformer version has the same idea. Fig. 13.5 shows the intuition of the architecture at a high level You'll see that the encoder-decoder architecture is made up of two transformers: an **encoder**, which is the same as the basic transformers from Chapter 10, and a **decoder**, which is augmented with a special new layer called the **cross-attention** layer. The encoder takes the source language input words $\mathbf{X} = \mathbf{x}_1, ..., \mathbf{x}_T$ and maps them to an output representation $\mathbf{H}^{enc} = \mathbf{h}_1, ..., \mathbf{h}_T$; usually via $N = 6$ stacked encoder blocks.

The decoder, just like the encoder-decoder RNN, is essentially a conditional language model that attends to the encoder representation and generates the target words one by one, at each timestep conditioning on the source sentence and the previously generated target language words to generate a token. This can use any of the decoding methods discussed in Chapter 10: greedy decoding in which case we generate the most probable token $\hat{y}_t$ (we'll use w to stand for tokens here):

$$\hat{y}_t = \text{argmax}_{w \in V} P(w|x, y_1 ... y_{t-1}) \tag{13.11}$$

or we can use beam search, or sampling methods like temperature sampling.

But the components of the architecture differ somewhat from the RNN and also from the transformer block we've seen. First, in order to attend to the source language, the transformer blocks in the decoder have an extra **cross-attention** layer. Recall that the transformer block of Chapter 10 consists of a self-attention layer that attends to the input from the previous layer, followed by layer norm, a feed forward layer, and another layer norm. The decoder transformer block includes an

cross-attention

extra layer with a special kind of attention, **cross-attention** (also sometimes called **encoder-decoder attention** or **source attention**). Cross-attention has the same form as the multi-headed self-attention in a normal transformer block, except that while the queries as usual come from the previous layer of the decoder, the keys and values come from the output of the *encoder*.

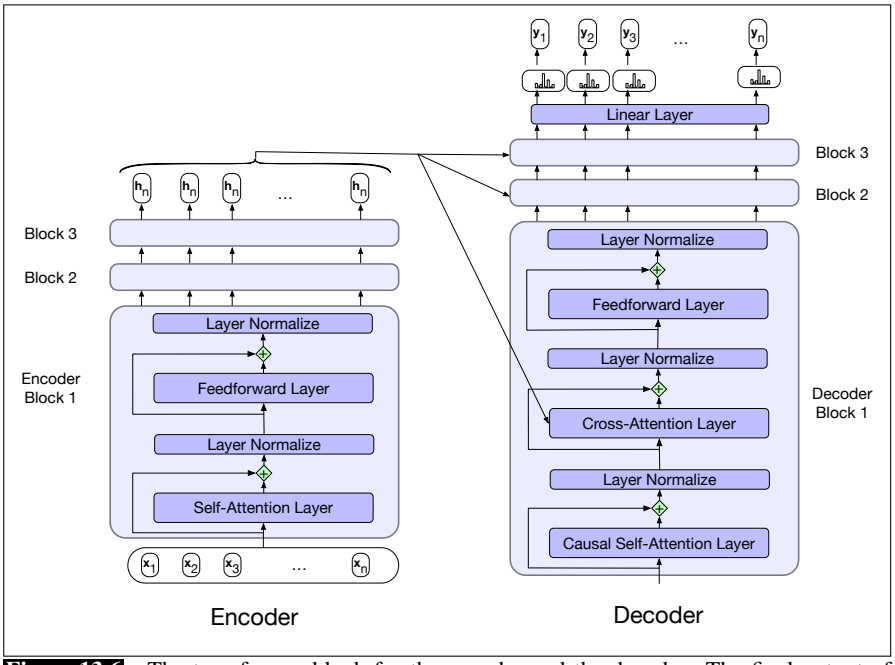


Figure 13.6 The transformer block for the encoder and the decoder. The final output of the encoder $\mathbf{H}^{enc} = \mathbf{h}_1, ..., \mathbf{h}_T$ is the context used in the decoder. The decoder is a standard transformer except with one extra layer, the **cross-attention** layer, which takes that decoder output $\mathbf{H}^{enc}$ and uses it to form its $\mathbf{K}$ and $\mathbf{V}$ inputs.

That is, the final output of the encoder $\mathbf{H}^{enc} = \mathbf{h}_1, ..., \mathbf{h}_t$ is multiplied by the cross-attention layer's key weights $\mathbf{W^K}$ and value weights $\mathbf{W^V}$, but the output from the prior decoder layer $\mathbf{H}^{dec[i-1]}$ is multiplied by the cross-attention layer's query weights $\mathbf{W^Q}$:

$$\mathbf{Q} = \mathbf{W^Q}\mathbf{H}^{dec[i-1]}; \;\; \mathbf{K} = \mathbf{W^K}\mathbf{H}^{enc}; \;\; \mathbf{V} = \mathbf{W^V}\mathbf{H}^{enc} \tag{13.12}$$

$$\text{CrossAttention}(\mathbf{Q},\mathbf{K},\mathbf{V}) \;=\; \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\intercal}{\sqrt{d_k}}\right)\mathbf{V} \tag{13.13}$$

The cross attention thus allows the decoder to attend to each of the source language words as projected into the entire encoder final output representations. The other attention layer in each decoder block, the self-attention layer, is the same causal (left-to-right) self-attention that we saw in Chapter 9. The self-attention in the encoder, however, is allowed to look ahead at the entire source language text.

To train an encoder-decoder model, we use the same self-supervision model we used for training encoder-decoders RNNs in Chapter 9. The network is given the source text and then starting with the separator token is trained autoregressively to predict the next token $\mathbf{y}_t$, using cross-entropy loss:

$$L_{CE}(\hat{\mathbf{y}}_t, \mathbf{y}_t) \;=\; -\log \hat{\mathbf{y}}_t[w_{t+1}] \tag{13.14}$$

teacher forcing

As in that case, we use **teacher forcing** in the decoder. Recall that in teacher forcing, at each time step in decoding we force the system to use the gold target token from training as the next input x_{t+1}, rather than allowing it to rely on the (possibly erroneous) decoder output $\hat{y}_t$.

13.4 Translating in low-resource situations

For some languages, and especially for English, online resources are widely available. There are many large parallel corpora that contain translations between English and many languages. But the vast majority of the world's languages do not have large parallel training texts available. An important ongoing research question is how to get good translation with lesser resourced languages. The resource problem can even be true for high resource languages when we need to translate into low resource domains (for example in a particular genre that happens to have very little bitext).

Here we briefly introduce two commonly used approaches for dealing with this data sparsity: **backtranslation**, which is a special case of the general statistical technique called **data augmentation**, and **multilingual models**, and also discuss some socio-technical issues.

13.4.1 Data Augmentation

Data augmentation is a statistical technique for dealing with insufficient training data, by adding new synthetic data that is generated from the current natural data.

backtranslation

The most common data augmentation technique for machine translation is called **backtranslation**. Backtranslation relies on the intuition that while parallel corpora may be limited for particular languages or domains, we can often find a large (or at least larger) monolingual corpus, to add to the smaller parallel corpora that are available. The algorithm makes use of monolingual corpora in the **target** language by creating synthetic bitexts.

In backtranslation, our goal is to improve source-to-target MT, given a small parallel text (a bitext) in the source/target languages, and some monolingual data in the target language. We first use the bitext to train a MT system in the **reverse** direction: a target-to-source MT system . We then use it to translate the monolingual target data to the source language. Now we can add this synthetic bitext (natural target sentences, aligned with MT-produced source sentences) to our training data, and retrain our source-to-target MT model. For example suppose we want to translate from Navajo to English but only have a small Navajo-English bitext, although of course we can find lots of monolingual English data. We use the small bitext to build an MT engine going the other way (from English to Navajo). Once we translate the monolingual English text to Navajo, we can add this synthetic Navajo/English bitext to our training data.

Backtranslation has various parameters. One is how we generate the backtranslated data; we can run the decoder in greedy inference, or use beam search. Or we can do sampling, like the temperature sampling algorithm we saw in Chapter 10. Another parameter is the ratio of backtranslated data to natural bitext data; we can choose to upsample the bitext data (include multiple copies of each sentence). In general backtranslation works surprisingly well; one estimate suggests that a system trained on backtranslated text gets about 2/3 of the gain as would training on the

same amount of natural bitext (Edunov et al., 2018).

13.4.2 Multilingual models

The models we've described so far are for bilingual translation: one source language, one target language. It's also possible to build a **multilingual** translator.

In a multilingual translator, we train the system by giving it parallel sentences in many different pairs of languages. That means we need to tell the system which language to translate from and to! We tell the system which language is which by adding a special token l_s to the encoder specifying the source language we're translating from, and a special token l_t to the decoder telling it the target language we'd like to translate into.

Thus we slightly update Eq. 13.9 above to add these tokens in Eq. 13.16:

$$\mathbf{h} = \text{encoder}(x, l_s) \tag{13.15}$$

$$y_{i+1} = \text{decoder}(\mathbf{h}, l_t, y_1, \ldots, y_i)) \quad \forall i \in [1, \ldots, m] \tag{13.16}$$

One advantage of a multilingual model is that they can improve the translation of lower-resourced languages by drawing on information from a similar language in the training data that happens to have more resources. Perhaps we don't know the meaning of a word in Galician, but the word appears in the similar and higher-resourced language Spanish.

13.4.3 Sociotechnical issues

Many issues in dealing with low-resource languages go beyond the purely technical. One problem is that for low-resource languages, especially from low-income countries, native speakers are often not involved as the curators for content selection, as the language technologists, or as the evaluators who measure performance (∀ et al., 2020). Indeed, one well-known study that manually audited a large set of parallel corpora and other major multilingual datasets found that for many of the corpora, less than of the sentences were of acceptable quality, with a lot of data consisting of repeated sentences with web boilerplate or incorrect translations, suggesting that native speakers may not have been sufficiently involved in the data process (Kreutzer et al., 2022).

Other issues, like the tendency of many MT approaches to focus on the case where one of the languages is English (Anastasopoulos and Neubig, 2020), have to do with allocation of resources. Where most large multilingual systems were trained on bitexts in which English was one of the two languages, recent huge corporate systems like those of Fan et al. (2021) and Team et al. (2022) and datasets like Schwenk et al. (2021) attempt to handle large numbers of languages (up to 200 languages) and create bitexts between many more pairs of languages and not just through English.

At the smaller end, ∀ et al. (2020) propose a participatory design process to encourage content creators, curators, and language technologists who speak these low-resourced languages to participate in developing MT algorithms. They provide online groups, mentoring, and infrastructure, and report on a case study on developing MT algorithms for low-resource African languages. Among their conclusions was perform MT evaluation by post-editing rather than direct evaluation, since having labelers edit an MT system and then measure the distance between the MT output and its post-edited version both was simpler to train evaluators and makes it easier to

measure true errors in the MT output and not differences due to linguistic variation (Bentivogli et al., 2018).

13.5 MT Evaluation

Translations are evaluated along two dimensions:

adequacy
1. **adequacy:** how well the translation captures the exact meaning of the source sentence. Sometimes called **faithfulness** or **fidelity**.

fluency
2. **fluency:** how fluent the translation is in the target language (is it grammatical, clear, readable, natural).

Using humans to evaluate is most accurate, but automatic metrics are also used for convenience.

13.5.1 Using Human Raters to Evaluate MT

The most accurate evaluations use human raters, such as online crowdworkers, to evaluate each translation along the two dimensions. For example, along the dimension of **fluency**, we can ask how intelligible, how clear, how readable, or how natural the MT output (the target text) is. We can give the raters a scale, for example, from 1 (totally unintelligible) to 5 (totally intelligible, or 1 to 100, and ask them to rate each sentence or paragraph of the MT output.

We can do the same thing to judge the second dimension, **adequacy**, using raters to assign scores on a scale. If we have bilingual raters, we can give them the source sentence and a proposed target sentence, and rate, on a 5-point or 100-point scale, how much of the information in the source was preserved in the target. If we only have monolingual raters but we have a good human translation of the source text, we can give the monolingual raters the human reference translation and a target machine translation and again rate how much information is preserved. An alternative is to

ranking
do **ranking**: give the raters a pair of candidate translations, and ask them which one they prefer.

Training of human raters (who are often online crowdworkers) is essential; raters without translation expertise find it difficult to separate fluency and adequacy, and so training includes examples carefully distinguishing these. Raters often disagree (source sentences may be ambiguous, raters will have different world knowledge, raters may apply scales differently). It is therefore common to remove outlier raters, and (if we use a fine-grained enough scale) normalizing raters by subtracting the mean from their scores and dividing by the variance.

As discussed above, an alternative way of using human raters is to have them **post-edit** translations, taking the MT output and changing it minimally until they feel it represents a correct translation. The difference between their post-edited translations and the original MT output can then be used as a measure of quality.

13.5.2 Automatic Evaluation

While humans produce the best evaluations of machine translation output, running a human evaluation can be time consuming and expensive. For this reason automatic metrics are often used as temporary proxies. Automatic metrics are less accurate than human evaluation, but can help test potential system improvements, and even

be used as an automatic loss function for training. In this section we introduce two families of such metrics, those based on character- or word-overlap and those based on embedding similarity.

Automatic Evaluation by Character Overlap: chrF

chrF

The simplest and most robust metric for MT evaluation is called **chrF**, which stands for **character F-score** (Popović, 2015). chrF (along with many other earlier related metrics like BLEU, METEOR, TER, and others) is based on a simple intuition derived from the pioneering work of Miller and Beebe-Center (1956): a good machine translation will tend to contain characters and words that occur in a human translation of the same sentence. Consider a test set from a parallel corpus, in which each source sentence has both a gold human target translation and a candidate MT translation we'd like to evaluate. The chrF metric ranks each MT target sentence by a function of the number of character n-gram overlaps with the human translation.

Given the hypothesis and the reference, chrF is given a parameter k indicating the length of character n-grams to be considered, and computes the average of the k precisions (unigram precision, bigram, and so on) and the average of the k recalls (unigram recall, bigram recall, etc.):

chrP percentage of character 1-grams, 2-grams, ..., k-grams in the hypothesis that occur in the reference, averaged.

chrR percentage of character 1-grams, 2-grams,..., k-grams in the reference that occur in the hypothesis, averaged.

The metric then computes an F-score by combining chrP and chrR using a weighting parameter β. It is common to set $\beta = 2$, thus weighing recall twice as much as precision:

$$\text{chrF}\beta = (1+\beta^2)\frac{\text{chrP}\cdot\text{chrR}}{\beta^2\cdot\text{chrP}+\text{chrR}} \tag{13.17}$$

For $\beta = 2$, that would be:

$$\text{chrF2} = \frac{5\cdot\text{chrP}\cdot\text{chrR}}{4\cdot\text{chrP}+\text{chrR}}$$

For example, consider two hypotheses that we'd like to score against the reference translation *witness for the past*. Here are the hypotheses along with chrF values computed using parameters $k = \beta = 2$ (in real examples, k would be a higher number like 6):

REF: `witness for the past,`
HYP1: `witness of the past,` chrF2,2 = .86
HYP2: `past witness` chrF2,2 = .62

Let's see how we computed that chrF value for HYP1 (we'll leave the computation of the chrF value for HYP2 as an exercise for the reader). First, chrF ignores spaces, so we'll remove them from both the reference and hypothesis:

REF: `witnessforthepast,` (18 unigrams, 17 bigrams)
HYP1: `witnessofthepast,` (17 unigrams, 16 bigrams)

Next let's see how many unigrams and bigrams match between the reference and hypothesis:

unigrams that match: `w i t n e s s f o t h e p a s t ,` (17 unigrams)
bigrams that match: `wi it tn ne es ss th he ep pa as st t,` (13 bigrams)

We use that to compute the unigram and bigram precisions and recalls:

unigram P:	17/17 = 1	unigram R:	17/18 = .944
bigram P:	13/16 = .813	bigram R:	13/17 = .765

Finally we average to get chrP and chrR, and compute the F-score:

$$\begin{aligned} \text{chrP} &= (17/17 + 13/16)/2 = .906 \\ \text{chrR} &= (17/18 + 13/17)/2 = .855 \\ \text{chrF2,2} &= 5\frac{\text{chrP} * \text{chrR}}{4\text{chrP} + \text{chrR}} = .86 \end{aligned}$$

chrF is simple, robust, and correlates very well with human judgments in many languages (Kocmi et al., 2021).

Alternative overlap metric: BLEU

There are various alternative overlap metrics. For example, before the development of chrF, it was common to use a word-based overlap metric called **BLEU** (for BiLingual Evaluation Understudy), that is purely precision-based rather than combining precision and recall (Papineni et al., 2002). The BLEU score for a corpus of candidate translation sentences is a function of the **n-gram word precision** over all the sentences combined with a brevity penalty computed over the corpus as a whole.

What do we mean by n-gram precision? Consider a corpus composed of a single sentence. The unigram precision for this corpus is the percentage of unigram tokens in the candidate translation that also occur in the reference translation, and ditto for bigrams and so on, up to 4-grams. BLEU extends this unigram metric to the whole corpus by computing the numerator as the sum over all sentences of the counts of all the unigram types that also occur in the reference translation, and the denominator is the total of the counts of all unigrams in all candidate sentences. We compute this n-gram precision for unigrams, bigrams, trigrams, and 4-grams and take the geometric mean. BLEU has many further complications, including a brevity penalty for penalizing candidate translations that are too short, and it also requires the n-gram counts be clipped in a particular way.

Because BLEU is a word-based metric, it is very sensitive to word tokenization, making it impossible to compare different systems if they rely on different tokenization standards, and doesn't work as well in languages with complex morphology. Nonetheless, you will sometimes still see systems evaluated by BLEU, particularly for translation into English. In such cases it's important to use packages that enforce standardization for tokenization like SACREBLEU (Post, 2018).

Statistical Significance Testing for MT evals

Character or word overlap-based metrics like chrF (or BLEU, or etc.) are mainly used to compare two systems, with the goal of answering questions like: did the new algorithm we just invented improve our MT system? To know if the difference between the chrF scores of two MT systems is a significant difference, we use the paired bootstrap test, or the similar randomization test.

To get a confidence interval on a single chrF score using the bootstrap test, recall from Section 4.9 that we take our test set (or devset) and create thousands of pseudo-testsets by repeatedly sampling with replacement from the original test set. We now compute the chrF score of each of the pseudo-testsets. If we drop the top 2.5% and bottom 2.5% of the scores, the remaining scores will give us the 95% confidence interval for the chrF score of our system.

To compare two MT systems A and B, we draw the same set of pseudo-testsets, and compute the chrF scores for each of them. We then compute the percentage of pseudo-test-sets in which A has a higher chrF score than B.

chrF: Limitations

While automatic character and word-overlap metrics like chrF or BLEU are useful, they have important limitations. chrF is very local: a large phrase that is moved around might barely change the chrF score at all, and chrF can't evaluate cross-sentence properties of a document like its discourse coherence (Chapter 27). chrF and similar automatic metrics also do poorly at comparing very different kinds of systems, such as comparing human-aided translation against machine translation, or different machine translation architectures against each other (Callison-Burch et al., 2006). Instead, automatic overlap metrics like chrF are most appropriate when evaluating changes to a single system.

13.5.3 Automatic Evaluation: Embedding-Based Methods

The chrF metric is based on measuring the exact character n-grams a human reference and candidate machine translation have in common. However, this criterion is overly strict, since a good translation may use alternate words or paraphrases. A solution first pioneered in early metrics like METEOR (Banerjee and Lavie, 2005) was to allow synonyms to match between the reference x and candidate $\tilde{x}$. More recent metrics use BERT or other embeddings to implement this intuition.

For example, in some situations we might have datasets that have human assessments of translation quality. Such datasets consists of tuples $(x, \tilde{x}, r)$, where $x = (x_1, \ldots, x_n)$ is a reference translation, $\tilde{x} = (\tilde{x}_1, \ldots, \tilde{x}_m)$ is a candidate machine translation, and $r \in \mathbb{R}$ is a human rating that expresses the quality of $\tilde{x}$ with respect to x. Given such data, algorithms like COMET (Rei et al., 2020) BLEURT (Sellam et al., 2020) train a predictor on the human-labeled datasets, for example by passing x and $\tilde{x}$ through a version of BERT (trained with extra pretraining, and then fine-tuned on the human-labeled sentences), followed by a linear layer that is trained to predict r. The output of such models correlates highly with human labels.

In other cases, however, we don't have such human-labeled datasets. In that case we can measure the similarity of x and $\tilde{x}$ by the similarity of their embeddings. The BERTSCORE algorithm (Zhang et al., 2020) shown in Fig. 13.7, for example, passes the reference x and the candidate $\tilde{x}$ through BERT, computing a BERT embedding for each token x_i and $\tilde{x}_j$. Each pair of tokens $(x_i, \tilde{x}_j)$ is scored by its cosine $\frac{x_i \cdot \tilde{x}_j}{|x_i||\tilde{x}_j|}$. Each token in x is matched to a token in $\tilde{x}$ to compute recall, and each token in $\tilde{x}$ is matched to a token in x to compute precision (with each token greedily matched to the most similar token in the corresponding sentence). BERTSCORE provides precision and recall (and hence F_1):

$$R_{\text{BERT}} = \frac{1}{|x|} \sum_{x_i \in x} \max_{\tilde{x}_j \in \tilde{x}} x_i \cdot \tilde{x}_j \qquad P_{\text{BERT}} = \frac{1}{|\tilde{x}|} \sum_{\tilde{x}_j \in \tilde{x}} \max_{x_i \in x} x_i \cdot \tilde{x}_j \tag{13.18}$$

13.6 Bias and Ethical Issues

Machine translation raises many of the same ethical issues that we've discussed in earlier chapters. For example, consider MT systems translating from Hungarian

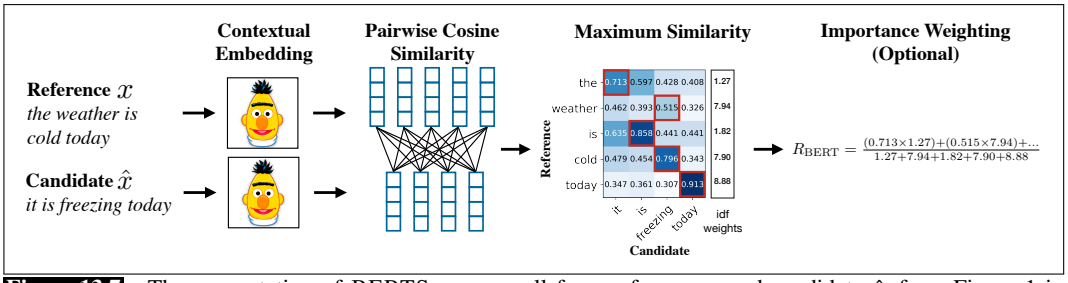


Figure 13.7 The computation of BERTSCORE recall from reference x and candidate $\hat{x}$, from Figure 1 in Zhang et al. (2020). This version shows an extended version of the metric in which tokens are also weighted by their idf values.

(which has the gender neutral pronoun *ő*) or Spanish (which often drops pronouns) into English (in which pronouns are obligatory, and they have grammatical gender). When translating a reference to a person described without specified gender, MT systems often default to male gender (Schiebinger 2014, Prates et al. 2019). And MT systems often assign gender according to culture stereotypes of the sort we saw in Section 6.11. Fig. 13.8 shows examples from Prates et al. (2019), in which Hungarian gender-neutral *ő is a nurse* is translated with *she*, but gender-neutral *ő is a CEO* is translated with *he*. Prates et al. (2019) find that these stereotypes can't completely be accounted for by gender bias in US labor statistics, because the biases are **amplified** by MT systems, with pronouns being mapped to male or female gender with a probability higher than if the mapping was based on actual labor employment statistics.

Hungarian (gender neutral) source	English MT output
ő egy ápoló	she is a nurse
ő egy tudós	he is a scientist
ő egy mérnök	he is an engineer
ő egy pék	he is a baker
ő egy tanár	she is a teacher
ő egy esküvőszervező	she is a wedding organizer
ő egy vezérigazgató	he is a CEO

Figure 13.8 When translating from gender-neutral languages like Hungarian into English, current MT systems interpret people from traditionally male-dominated occupations as male, and traditionally female-dominated occupations as female (Prates et al., 2019).

Similarly, a recent challenge set, the WinoMT dataset (Stanovsky et al., 2019) shows that MT systems perform worse when they are asked to translate sentences that describe people with non-stereotypical gender roles, like "The doctor asked the nurse to help her in the operation".

Many ethical questions in MT require further research. One open problem is developing metrics for knowing what our systems don't know. This is because MT systems can be used in urgent situations where human translators may be unavailable or delayed: in medical domains, to help translate when patients and doctors don't speak the same language, or in legal domains, to help judges or lawyers communicate with witnesses or defendants. In order to 'do no harm', systems need ways to assign **confidence** values to candidate translations, so they can abstain from giving incorrect translations that may cause harm.

confidence