Chapter 8

Custom machine translation

Gema Ramírez-Sánchez
Prompsit Language Engineering

This chapter gives an overview of the theoretical and practical implications of customizing machine translation (MT) to make it fit for a particular purpose. The chapter is written for readers who have just a basic knowledge of MT, but experts who are seeking new ways of explaining MT to non-experts may also find it useful. The MT paradigm assumed in the chapter is that of neural MT.

1 Introduction

1.1 Generic machine translation

Most casual users of machine translation (MT) undoubtedly rely on *generic* MT, that is, MT based on engines trained to cover a wide range of topics, styles and genres, and not specialized in any particular domain.

While generic engines may be perfectly suitable for general-purpose usage, they may become less useful for texts that use a narrow range of vocabulary, or have very particular, characteristic styles, or are constrained by the conventions of a particular genre. This typically applies to texts associated with highly specialized domains such as law or medicine, but such constraints are also a feature of texts that we encounter in every-day life. Recipes, for example, have typical structures and vocabulary that differentiate them from other "every-day" texts like consumer guides. You hardly ever find questions in recipes, but these are frequent in consumer guides. Both types of text are often translated into other languages, either for casual use (think of a search engine translating something like "how can I make chocolate cookies?", or "what type of light bulb is recommended to save energy?"), or for professional use (think of a publisher translating a recipe book, or a manufacturer translating technical specifications for a

Gema Ramírez-Sánchez. 2022. Custom machine translation. In Dorothy Kenny (ed.), *Machine translation for everyone: Empowering users in the age of artificial intelligence*, 165–186. Berlin: Language Science Press. DOI: 10.5281/zenodo.6760022

consumer product). In the following paragraphs, I use examples from a recipe (for apple crumble) and a consumer guide (for a type of light bulb) to see how generic MT engines cope when faced with the specific terminology used in these two every-day genres.

Recipes are frequently divided into three parts: title, ingredients and instructions. For generic MT, a simple recipe title can already be a struggle. Table 1 shows how three generic MT systems cope with the term *apple crumble* in Spanish, French and Italian. MT1 works well for French and Italian, but not Spanish; MT2 does not work well in any target language; and MT3 translates adequately into French, but poorly into Spanish and Italian.

Table 1: Machine Translation of a recipe title: apple crumble.

	MT1	MT2	MT3
Spanish	migas de manzana 'crumbs of apple'	se desmorona la manzana 'the apple falls apart'	Desmoronamiento de la manzana 'falling apart of the apple'
French	crumble aux pommes 'apple crumble'	Crumble d'apple 'crumble of apple'	Crumble aux pommes 'apple crumble'
Italian	Crumble di mele 'apple crumble'	La mela si sbriciola 'the apple crumbles'	Crumble di mela 'crumble of apple'

Don't worry if you are not fluent in Spanish, French or Italian. Just do a similar test yourself by translating *apple crumble* into a language you know using your favourite online MT service. You'll probably see translations related to 'collapsing apples' or 'apples that fall apart' like we see in some of the examples in Table 1. Other translations, namely *crumble aux pommes* and *Crumble di mele*, are good (which is why we have glossed them in Table 1 simply as 'apple crumble'). How the MT engine copes depends on the data used to train the engines. Special steps may also be taken to feed engines with the correct terminology during training or as a post-translation step. But, without built-in treatment of specialized terminology, what we get from system MT2 is fairly typical of what we can expect from generic MT.

Now let's dive into the intriguing world of light bulbs. A huge variety of specialized information is available to consumers eager to learn about the many types of light bulbs on the market. So, imagine you are a non-native speaker of English living in an English-speaking country; a light goes out, and your neighbour offers to give you a twisted fluorescent lamp. You offer your best smile in exchange, as you are not sure what the neighbour means exactly. Using your phone, you check the translations provided by some generic MT systems. They provide the translations reproduced in Table 2.

Table 2: Machine Translation of a type of light bulb: twisted fluorescent lamp.

	MT1	MT2	MT3	Should be
Spanish	Lámpara fluorescente retorcida	Lámpara fluorescente retorcida	Lámpara fluorescente retorcida	Bombilla fluorescente en espiral
French	Lampe fluorescente torsadée	Lampe fluorescente tordue	Lampe fluorescente torsadée	Ampoule spirale fluorescente
Italian	Lampada fluorescente contorta	Lampada fluorescente contorto	Lampada fluorescente attorcigliata	Lampadina fluorescente spirale

After reading this, you expect your neighbour to give you a funny-shaped standard lamp or table lamp. Where are you getting this idea from? Oh, ambiguity: none of the engines provides a word meaning *light bulb* as a translation for *lamp*. All of them go for the other meaning of *lamp*, where the word stands for the whole piece of *lighting equipment*.

Generic MT got it wrong, but a custom MT engine should be able to get it right. But what is custom MT? The next section should give you an idea.

2 Custom machine translation

Custom MT, as opposed to generic MT, is MT that is designed to fit a specific purpose.

Imagine you work for a company that produces a big car brand. Like other manufacturers in the automotive sector, your company will produce lots of technical and user manuals as well as marketing material in dozens of languages.

Anything that helps your company improve communication with its internal staff, train car salespeople, or convince buyers, is deemed a key activity, and in a multilingual environment like this one, MT can be very helpful. Your company thus uses MT to produce the first draft of nearly all its translations. Reviewers, known as post-editors (see O'Brien 2022 [this volume]), then improve these drafts.

Your company starts out on its MT journey using a generic MT system and improving the output through post-editing. Soon, the post-editors begin to realize that they need to fix the same terminology, genre and style mistakes over and over again: this is not very appealing or efficient. Your company then remembers that it has been producing translations for decades, and wonders whether it could use these existing translations to somehow improve the process.

The answer is yes. But how? First, by including its own past translations as training data, your company allows the MT engine to learn from them. That is, it uses its own training data to create a custom MT engine. The custom engine produces draft translations which are much closer to the company's past translations and have far fewer errors in terminology and style, and the post-editors are happier.

But are things really as simple as that? Well, yes, but only if you have a sufficient amount of data (millions of translated sentences) which is in the right format (aligned parallel data; see Kenny 2022 [this volume]), is internally consistent (otherwise be prepared for inconsistent output), and is in the desired language pair or pairs. You also need engineers or external providers to train the system and integrate it into the company's translation workflow, as well as the right hardware and software. All this just to start with. Then you need a retraining plan if you want to take advantage of the next translations that will be produced: this can be on-the-fly if you work with adaptive MT, every six hours if you have crazy production numbers, every six months or once a year if you just want to keep the system up-to-date and consistent.

So maybe "simple" is not the word, and you might ask whether all the effort will be worth it? Let's set reasonable expectations.

2.1 What can we expect from custom machine translation?

Once the preserve of MT experts, custom MT is now commonly encountered by all sorts of users. It is even shown to us raw, without any revision, and is performed on-the-fly when we click a "get a translation" button. We see it on online hotel booking websites, online technical support for specialized software, job vacancy listings, teachers' messages in educational apps, etc.

As general users, our main goal is usually to understand information we have retrieved from somewhere, often a website. In such cases, we might expect that custom MT will, at the very least, do a better job in outputting accurate terminology and idioms than generic MT would; and we might also expect better adherence to text style.

For example, if we are looking for information about nursing homes on a website, we expect *home* to be translated to communicate the sense 'main page' on the navigation menu but 'institution where people are cared for' in the website content. If we are looking at a baseball website, there should be no confusion between the 'main page', 'base' or 'team's own grounds' senses of *home*. In other contexts, *home* should be not translated at all; for example, when it is part of a brand name. Better management of specific translations for specific contexts is something that we can expect from custom MT.

For language professionals, other, seemingly small, details take on particular importance. They appreciate when their custom MT engine outputs the right upper and lower case forms of words, or can cope with formatting conventions like bulleted lists, or renders numbers appropriately. When these phenomena are not handled correctly (as in Table 3), post-editors need to review the output of the MT engine carefully and correct these "little" mistakes, which is both irritating and time-consuming. Adherence to textual conventions and style is also something expected in a professional environment.

Table 3: Machine translation output details matter

Input	Output: generic MT		
English	Spanish	French	Italian
For this match, the following players will be excluded a) One b) Four c) 6	Para este partido, los siguientes jugadores serán excluidos: uno b) cuatro c) seis	Pour ce match, les joueurs suivants seront exclus: un b) Quatre c) Six	Per questa partita, saranno esclusi i seguenti giocatori: un solo b) Quattro c) Sei

Given sufficient effort and the right resources, custom MT is capable of outputting text without the kinds of error in Table 3. Among the resources required are suitable human resources, which I address in the next section.

2.2 Who customizes machine translation?

As you will learn from this short section, MT is a truly interdisciplinary field in which computer engineers work alongside linguists.

Back in the days when linguistically-aware, rule-based MT was state-of-the-art, both engineers and linguists had an active role in building systems: the linguists wrote the grammatical rules and dictionaries, and the engineers wrote the computer programs that implemented the rules.

Later on, in statistical MT times, engineers took almost full responsibility for the process. Linguists were occasionally involved in output evaluation, but hardly ever in error analysis and never in defining action points to improve the machine translated output.

The technology was still reliant on human translators, however, as they provided the translations that fed the translation models.[1] But I can confirm that, in the heyday of statistical MT, linguists were more or less excluded from the actual building of the systems.

Nowadays, little by little, linguistics-savvy engineers have started to pay attention not only to tweaking parameters and hardware or automatic evaluation metrics, but also to the quality of the translations produced by their systems, at a very fine-grained level. At the same time, technology-savvy linguists have started to get involved in assessing and curating the data needed to train engines, in playing with training kits, in evaluating systems, and in defining a strategy not only to improve them but also to integrate them into the translation workflows of client companies.

The ideal situation given the nature of the task is collaboration between professionals from the two fields, professionals who are interested in learning about and contributing to each other's areas. This is particularly effective for custom MT where the need to tune systems to a particular domain means that there are significant benefits to be gained from linguists assessing the usefulness of the training data, taking control over (or at least understanding) the strategy followed during the training process, and analyzing the output. Custom MT also benefits from engineers understanding the specifics of the texts and the languages they are working with in order to be creative in finding solutions to advance and solve the main issues present in the output: is a new module or a pre- or post-processing step needed to cope with rich morphology, product names or alphanumeric codes, for example?

[1] Keep this in mind next time you hear that "machine translation is achieving human parity *without human intervention*". So, what about the texts used to train the engines?

Training for linguists and engineers in the more technical or linguistic aspects of MT is becoming more and more usual in educational environments, but it is still predominant in professional contexts, where needs arise in a more dynamic way.

Before concluding this section, it is worth mentioning another group of people, who are increasingly contributing to MT customization, but who may or may not be covered by the term *linguists* used above, namely *translators*. In this chapter I mainly cover custom MT as an off-line activity which requires human intervention and which happens only from time to time. However, it is worth noting that in some current settings, namely those involving so-called "adaptive" MT (see O'Brien 2022 [this volume]), customization may also happen in real-time (or at least much faster and more frequently than in other settings). In such scenarios, translators and their translations are becoming the cornerstone for custom MT where customization actually means real-time user mimicking. New translations are integrated automatically into already proficient systems as the translator delivers new parallel sentences (source-target pairs), and preferred translations are made available to translators as they work.

3 How to customize a machine translation engine

3.1 An allegory

Imagine that you land on a new planet – no life found, but there is an extraordinary library with plenty of written texts in what seem to be a number of different languages: some texts are in one language (L1), some in another language (L2), some appear to be bilingual texts (L1-L2). But there is no grammar book, no orthography book, no organized linguistic knowledge; no clue as to how these languages work at all. Just texts with plenty of sentences, pairs of sentences and more sentences.

By chance, you also find lists of L1-L2 word correspondences. Also, while inspecting the texts, you observe that some of them are marked with a particular stamp, others are marked with a different stamp, and yet others do not have any stamp at all.[2] You feel lucky: all these texts are probably the only knowledge about the lost life on this newly discovered planet.

But no! Wait! As you leave the library, you discover that there is still life on this planet: two of its inhabitants are staring at you in a hostile way but you also

[2]For the purposes of this analogy, each different stamp represents a different domain, and texts with no stamp can be considered as non-domain specific.

notice that they look at each other in an even more inimical way. After some time, you discover that their lack of friendliness is due to a key factor: they don't understand one another; one is an L1 speaker, the other an L2 speaker and they did not know about the library. You need to help them! As a polyglot, you've done this before, and you have no option: you will teach them translation from L1 to L2 and the other way round.

How will you do this? By trying to learn about the languages individually first and then diving into how to translate between them? (This will take too long.) Or by going directly to the bilingual texts and lists? (This seems more promising.) You might start with the list of words and observe further relationships between other words, phrases and longer chunks. Or you could use the fact that texts can be classified into those with stamps and those without stamps. You could, for example, start grouping together texts carrying similar stamps. Or you could use all the texts at the same time. A myriad of possibilities is open to you to start learning from data about translation between L1 and L2.

At this point, when an MT system is learning how to translate, it is in the same situation as you are on this mission: you both have texts, bilingual (and monolingual), maybe also terminology lists, but nothing else. These are the only sources to learn from, but there are different ways of carrying out the learning process.

Back on your newly discovered planet, you first open your mind and try to learn from what you can observe in the bilingual L1-L2 texts. You use the lists already compiled to check if your assumptions are right and then try to build on these assumptions by forming new assumptions. You soon move from words to longer chunks. At this point you don't pay attention to whether a text is stamped or not; you try to use all resources together as a whole.

In a similar way, to build a first MT system, one usually starts by concatenating, or stringing together, all bilingual data, regardless of the different domains they come from, and by performing initial training using the default software settings.

After a first effort, you start getting messages from the L1 speaker and translating them into the L2. Then you show your translated messages to the L2 speaker to validate them. And then you repeat the process working the other way round. You keep improving your knowledge as you interpret the expressions on the L1 and L2 speakers' faces. Sometimes they laugh, but most of the time they nod their heads, and sometimes they even look as if they get it. You learn from their feedback and keep going.

In MT development, evaluation is not usually based on human (or extraterrestrial) assessment. Rather, we use automatic metrics to compute quality scores

based on a comparison of the machine's output with translations already produced by professional translators (see Rossi & Carré 2022 [this volume]). In most cases, the more similar the machine output is to the human translation, the better it is deemed to be. If the automatic metrics suggest that things are OK, and a quick inspection of the output suggests that it does not present any major issues, the system can be considered as a functional baseline. Otherwise, we keep training, maybe adding some pre-processing or post-processing. After each round of training, we check our automatic metrics. When the scores are as good as we think we can get them, we stop training.

Back on the newly discovered planet, as you progress in your learning process, you discover that there is more than one translation for some words in the same language combination, but that correspondents are consistent across texts marked with the same stamp. So you decide to separate the texts by their stamps and to compile specific correspondences in separate lists. You start further inspecting non-stamped texts and then move to stamped ones. Stamped texts look a bit different to non-stamped ones: for example, sentences tend to be very long in some stamped texts while in non-stamped texts they are very short.

Given this situation, depending on how much data we have and the final goal of the system, we could train an MT engine using only texts that share the same stamp (and where the stamp represents a domain). Our in-domain system could use both generic and domain-specific texts or just domain-specific ones. And we will definitely make the most of state-of-the-art MT techniques to make the system as domain-aware as possible. This is exactly what customization is about: playing with data and techniques. In what follows we explain each of these approaches in basic terms. A more comprehensive survey of *domain adaptation* in neural MT is provided by Saunders (2021).

3.2 Customization through data

MT can be adapted to fit a specific purpose by using specific texts: we can build a very good MT system for mobile phone descriptions provided that we have a sufficient number of texts that describe mobile phones and that they are translated into the language we want to target. We can also use monolingual texts or bilingual vocabulary lists that are specific to the domain in question. This is what we call *in-domain data*. The ideal situation is to have access to bilingual in-domain data in the form of parallel sentences.[3]

[3]Pairs of source-target sentences that are translations of each other, and are ideally ordered as they appeared in the texts they came from, to take advantage of the co-text in recent sentences (see Kenny 2022 [this volume]).

3.2.1 How much data do we need?

It is difficult to say how much data is enough. For generic MT, the answer would be take as much as you can get, then maybe filter it based on qualitative factors, discarding, for example, very repetitive sentences (where there is not much to learn from), very ugly sentences (e.g. ones made up mostly of numbers), or very long sentences (which are too difficult to learn from). For custom MT, the answer could be the same, but this time also taking into account that in-domain data needs to represent a generous proportion of the whole training data set, otherwise our system will not be able to learn how to produce in-domain translations.

The unspecific measure "a generous proportion" is used here on purpose as we know that it is very rare to have enough in-domain data to train a system. After all, we will need at least several million sentence pairs; maybe less than for generic MT, but still a lot. So, we normally end up mixing the available in-domain data with generic or out-of-domain data.

Depending on the language combination, when adding the available in-domain data to the out-of-domain data as the first step in customizing an MT system, we are usually faced with one of two very different scenarios: we either have too much data or too little data. When it comes to data, size matters.

3.2.1.1 A "too much" data scenario

A scenario in which we have too much data can lead us to impractical situations in which we need an unreasonable amount of time and number of servers up and running to train systems. It is difficult to give a precise figure, so let's say that you have too much data when you realize that with less data, you get the same results and, as a bonus, you need fewer computational resources and less time to achieve them. (For more details on how developers use metrics such as BLEU to tell if an engine has stopped learning, see Pérez-Ortiz et al. (2022 [this volume]), especially 7.2).

In custom MT, the available in-domain data is prioritized and we use all of it. For the out-of-domain data, however, we will need to select a subset of the available parallel sentences. Data selection is normally performed using the in-domain data as a model of what we want. Selection can be done automatically in many different ways, for example by:

- Scoring the out-of-domain sentences by textual, semantic or syntactic similarity to the in-domain ones

- Grouping the out-of-domain data by the topics discovered in the in-domain data
- Re-classifying the out-of-domain sentences as in-domain based on examples of what would be very good or very bad in-domain sentences

3.2.1.2 A "too little" data scenario

In the opposite "too little" data scenario, we could compromise the usefulness of the MT system if we do not make an effort to get more data. The system would probably output poorly translated sentences with many untranslated words. We should start with the in-domain data, and try to extend it as much as possible. This extension can be done by integrating more already existing data (if available) or by creating it. We can get as-is or made-to-measure data for free or consider purchasing it. Automatic data extension strategies include:

- Getting additional bilingual data by crawling multilingual websites for the targeted languages in general, or paying specific attention to the vocabulary covered by the in-domain data
- Using additionally available monolingual data, ideally in the target language, and translating it back into the source language with a third-party MT system
- Pivoting through another language, translating monolingual data in two steps: first into the pivot language then into the target/source language of our combination
- Creating new sentences in the source and target languages by automatically composing new ones from the available data (replacing words by synonyms or similar frequency words, using automatic paraphrasing, etc.)

Experience shows that not only bilingual data, but also monolingual and multilingual data, and generic, in-domain and multi-domain data, have all proven useful in helping MT systems to learn (Saunders 2021). What is more, tiny amounts of data are starting to be taken into account in adaptive or incremental MT scenarios (see O'Brien 2022 [this volume]). The landscape is changing fast but one thing is certain: provided that there is some data, there is a chance for learning, and MT will make the most of it.

3.2.2 Data quality

Data quality also plays a role in the customization of MT systems, impacting directly on the quality of the final output. This has become a topic of interest particularly with the rise of neural MT (see Kenny 2022 [this volume] and Pérez-Ortiz et al. 2022 [this volume]), as studies show that it is very sensitive to noise in the training data (Khayrallah & Koehn 2018). Most of the work done to overcome this problem consists in filtering out noise using a mix of patterns and rules to remove obvious noise, scoring sentences for quality, and classifying them to discriminate between high-quality and low-quality content. It also includes the removal of duplicates (Khayrallah & Koehn 2018).

3.2.3 Data organization

Finally, data organization is also becoming a topic of interest (Mohiuddin et al. 2022). Some studies organize training data using sentences with similar length to improve training and translation speed. Others feed the models with sentences from more simple to more complex to improve quality. Others use documents instead of shuffled sentences to take advantage of the wider textual context, in order to get improved MT outputs.

3.3 Customization through techniques

Once we have compiled and cleaned all the data we can get our hands on to train an in-domain MT system, how do we use these data? What type of system architecture is best for our purposes? Do we have different options to constrain the output? This is what customization through techniques is all about. The techniques in question may have to do with modifying the architecture of the network, adjusting parameters during training or combining different systems during training or translation time (also called inference).

Below I review some of the most popular techniques to get domain-specific MT. Koehn (2020: Ch. 13) provides a more detailed discussion.

3.3.1 Self-taught systems

In the self-taught systems approach, we would use first only the generic data to train a system. Then, taking this system as a starting point, we would perform a second training pass using only the in-domain data. This would result in a fine-tuned system with some general knowledge of the languages and a very specific knowledge of the in-domain vocabulary, structures, etc.

3.3.2 Coached systems

Besides training the MT system using generic and in-domain data, one can also use language models trained ideally on in-domain texts in the target language, but at the very least good quality monolingual texts. The language model will help in tailoring the output towards more in-domain-like language. It is also possible to use domain controllers or discriminators to label training data at word, sentence or even embedding level. This technique consists in identifying precisely what in the generic data is closer to or further from the in-domain data and use this information during training.

3.3.3 Partnering systems

Some systems are based on ensembles of several components (e.g. domain-specific sub-networks) or even full systems. They combine their knowledge to produce better output together than individually.

4 Customization in practice

Theory and practice are frequently two sides of the same coin. This section gives very practical details on customizing a neural MT engine. It is aimed at beginners and does not assume advanced technical knowledge.

4.1 Available tools

There are already several professional kits for MT customization. These mostly allow users to play with data but not techniques. So while customization through data is well within the reach of a wider range of language professionals, customization through techniques remains the preserve of researchers and developers.

Most MT providers offer remote access to pre-trained generic or domain-specific engines for a given price. Some also offer customization options.[4] When customization is offered, this usually covers:

- Adding your own corpora
- Adding your own terminology

[4] At the time of writing, there are more than 40 providers offering MT services and around 20 provide some customization options. Source: https://inten.to/mt-landscape/, last accessed 26 June 2022.

- Training a new model with your data
- Testing the custom model

All this can be done in a semi-automatic way, where a real person takes care of the data and the processes involved, or in an fully automatic way, where customization happens without further human intervention.

There are also MT testing environments designed for teaching language professionals how MT works. These are used in translation technology classrooms or in professional environments as training tools. They usually offer customization options to see what happens when a system is trained with more generic or in-domain data. A good example of such an environment is MutNMT.[5]

4.2 Key factors when defining a strategy

The most important factors that need to be taken into account when defining a custom MT strategy are:

- Language combination
- Domain
- Available data
- Purpose of the system
- Deadline
- Hardware characteristics and availability

Let's take a look at each in turn:

Depending on the language combination, you may decide to work with one or another tool. Morphologically-rich languages, for example, may benefit from pre- or post-processing using a tool designed to support the language in question.

Some domains have very special textual characteristics that might be supported by training kits. For example, a particular training kit might be ideal for dealing with very long or very short sentences, numerical expressions, or proper names, which usually need to be retained in translation. If the domain has other very specific characteristics, they need to be taken into account.

[5] See http://www.multitrainmt.eu/index.php/en/neural-mt-training/mutnmt, last accessed 26 June 2022.

The amount and quality of available data is a key factor in deciding when to go ahead and train a system, or when to try customizing a system using more and/or higher quality data. Also, if there is data beyond parallel sentences that you want to include, it is necessary to make sure that technology will support it.

The purpose of most neural MT systems is to produce the best raw output possible, but additional requirements usually need to be met. Before you opt for a particular service provider you might need to consider whether their language models can be accessed remotely or whether you actually need access to them on your own premises. Will the system be used online with concurrent users or as a batch one-at-at-time queued process? Will the system translate text strings? Or does it need to support translation using different file formats? Will it be accessed through an API, web app, or a connector to a third-party tool? And so on.

Deployment of custom MT might require a trade off between quality and meeting a delivery deadline. The best possible system may need more training days than you can afford.

Finally, hardware is also a key component, both for training and for later use of a system in production. Depending on other factors, you may need a service that is available 24/7, and uses several GPUs/CPUs at the same time.

4.3 Getting the right data

Custom MT systems heavily depend on the availability and quality of in-domain parallel data that is similar to the texts we ultimately wish to translate. In 3.2.1, I discussed data sizes and ways to select or extend such data depending on the scenario. Here I cover a very basic question: where can you find the right parallel data for your system?

In the first instance, you might try getting existing parallel data for free: there is a myriad of data repositories that offer free and publicly available bilingual corpora ready to use for MT. Most of these repositories or corpora can be accessed through the OPUS website, which is maintained by the University of Helsinki, and hosts corpora in more than 200 language combinations.[6]

Purchasing parallel data is also an option. (Selling parallel data for MT is a business.) What is on offer varies from very large to very small collections, each with different usage rights, and using different business models and pricing. One of the largest MT-specific data collections is offered by TAUS and covers more than 600 language combinations.[7]

It is also possible to assemble parallel data yourself. Although probably not ideal for assembling large amounts of data, parallel corpora can be created by:

[6] See https://opus.nlpl.eu/index.php, last accessed 26 June 2022.

[7] See https://www.taus.net/, last accessed 26 June 2022.

- crawling multilingual websites: if a URL contains bilingual content, you can download such content for a given language pair, and either align it yourself (see Kenny 2022 [this volume] for an example of an aligned text), or use a third-party service to align it.
- aligning your own translated documents: if you have translated documents and their original source texts, you can use standard open-access or proprietary tools to align them at sentence level.[8]

4.4 Getting the data right

Once we've got the right data, we need to prepare it before training our system. This means making sure that we have:

- One text file per language (or language combination): plain text files are the usual food for MT training. One per language is the ideal, although some systems can cope with spreadsheet-type formats that use one column per language, or even TMX files (see Kenny 2022 [this volume]).
- One sentence per line: the text files need to be formatted so that each sentence takes up a single line. This effectively means that each sentence becomes a one-sentence paragraph. Longer or shorter units are not usually allowed,[9] although headings can be treated as if they are sentences.
- One-to-one aligned sentences: independently of the format, every line in the source file needs to correspond to the same line in the target file.
- Clean data: there should be no duplicates, typos, or noisy sentences (e.g. those with only numbers, or that are badly encoded, or not in the language you want, etc.).
- Anonymized data (if necessary): you might need to eliminate any sensitive data, and especially personal data, from your training corpora.
- Organized data: corpora need to be divided into three sets corresponding to the different phases of the training process. These sets are usually named the *training set* (*train* for short), the *validation set* (also *development set* or *dev* for short) and the *test set* (or just *test*).

[8]Free alignment tools include LF Aligner (https://sourceforge.net/projects/aligner/, last accessed 26 June 2022); while well-known paid-for alignment tools include those provided with translation memory tools (see Kenny 2022 [this volume]).

[9]I am assuming here sentences as the training unit, not documents.

Some rules of thumb for organizing data are as follows:

- Overlap between the sentences in these three sets must be avoided at all costs. Indeed, if possible, sentences from different sources should be used in each set to guarantee their balance and independence.
- Training sets can vary in size from several thousand to several million sentences, but validation and test sets do not normally exceed 5,000 sentences.
- Training data may contain generic and custom data, but validation and test data should be as in-domain as possible to test the suitability of the trained model for its intended purpose.
- When the training set contains both generic and in-domain data, the proportion of in-domain data needs to be as large as possible, otherwise the model will take most of its knowledge from the generic data.

Thus far, nothing we have mentioned is peculiar to custom engines; all this preparation applies equally to generic and custom MT. The next steps, related to pre-processing, also apply in both scenarios but may vary for particular languages or language combinations:

- Text needs to be tokenized: you need to provide the training process with texts where tokens can be clearly identified. Tokens are the different units into which one can divide a text. They can be words, spaces or punctuation marks, for example. Tokenizing is mostly about identifying word boundaries, and this sometimes involves guessing where a word starts and ends, often using the spaces between written words to guide us. Languages like Thai are not written with spaces between words however, which makes word tokenization challenging. Tokenizers for these languages sometimes simply split sentences into chunks containing seemingly arbitrary sequences of characters. This approach has the advantage of being language independent, even if it has no concept of "units of meaning".
- Text may need to be truecased: we may want our training process to capture the fact that a word spelled with an initial capital letter at the beginning of sentences in our training data (for example, *The*) is the same word as that spelt all in lower case (in this case, *the*) in other sentence positions in the data. We thus use *truecasers* to convert all words except proper names

(in languages like English) to lower case.[10] Truecasing only applies to languages that distinguish uppercase and lowercase so it is not applicable to Chinese, Arabic, Hebrew and many others.

- Sub-word splitting: depending on the tokenization performed on a text, further splitting of words into sub-words, characters or other chunks of text may apply. This splitting may be based on frequency counts or a linguistically-motivated segmentation that takes into account morphemes, stems, and additional morphological information. This is not always configurable in custom environments as it requires the neural network to be able to cope with special types of input and output, and sometimes only one method is supported in pre- or post-processing.

So, what if you have texts that you want to use as training data, but they are not in the right format? Don't worry, because most of the time, tokenization, truecasing and subword splitting – if applicable – are included by default as pre-training, pre-processing (and post-processing) steps in standard training kits. They are mentioned here so that users of these training kits know what's going on. There are also plenty of stand-alone tools you can use to perform these steps. To find these tools, you can go to platforms like Hugging Face[11] or Github,[12], or simply open a search engine and start typing: sentence splitter, sentence aligner, parallel corpus filtering, anonymizer, tokenizer, truecaser, etc. For more refined searches, add the source and target languages. You will discover a whole world of tools.

4.5 Training the custom model

Training an MT model once we have the right data, in the right format, is just a matter of clicking a button in many current training environments. Some environments allow users to tweak a number of parameters to get the best out of the combination of data-training environment and system architecture. Sometimes users are allowed to tweak settings for educational or research purposes.

The most common parameters that can be adjusted by the user before training are as follows:

- *Vocabulary size* specifies the number of different words or sub-words (also called sub-word units, types or word types) allowed in the vocabulary computed from the training corpus.

[10]Note that all nouns in German begin with upper case, and, like proper nouns in English, these should not be truecased.

[11]https://huggingface.co/, last accessed 26 June 2022.

[12]https://github.com/, last accessed 26 June 2022.

- *Batch size* is the number of tokens[13] that will be processed together in each training step. This is needed because it is not possible to feed all the data in the training set to the neural network at once.
- *Beam size* is the number of translation hypotheses (i.e. translation candidates) that are taken into account when translating a word. Hypotheses are produced during the training process and when the system is actually translating.
- *Duration* refers to the number of epochs allowed in the training process. An *epoch* is a full training pass over all the sentences in the training set. Each epoch comprises all the training steps necessary to see all the data once.[14]
- *Validation frequency* is the number of steps included before each evaluation of the status of the training takes place. Typically, validation cycles happen many times in each epoch.
- *Stopping condition* specifies the maximum number of validation cycles allowed where no improvement of the engine's performance is registered. When this maximum is reached, the training process is ended, regardless of the number of epochs initially set. Improvements can be measured based on any automatic metric, or on a combination of metrics. Common automatic metrics include as BLEU, chrF1 and perplexity.[15]

All of these parameters usually come with default values that developers have set after optimizing the training process for a particular environment.

Once parameters have been set – or the default parameters have been accepted – training proceeds as follows: at each training step, a batch of training data is fed into the neural network, the output for each sentence in the batch is computed, the error loss is computed, weights are updated, and it all starts again!

[13] Batch size in *tokens* (see Rossi & Carré 2022 [this volume]) instead of sentences has become the most used batch type in the last years in order to make batches more similarly sized.

[14] Given the large amounts of data used in NMT, the use of epochs to measure the duration of the training can be impractical. Rather than using epochs, you can use the number of steps, in relation to a particular batch size, to help you measure duration independently of the model, language pair or amount of data.

[15] Perplexity in natural language processing, and more specifically in MT, measures how uncertain a translation model is about predicting the next word when translating. A low perplexity is obtained when the translation model assigns a high probability to each word/token in a given target sentence. For more information on BLEU and chrF1, see Rossi & Carré (2022 [this volume]).

After a predetermined number of training steps (set by the validation frequency), the engine's performance is evaluated, and then training resumes. When further training fails to improve the engine's performance, or the performance starts to degrade, the training stops. There it is, our model!

4.6 Testing the custom model

Once we've trained an MT system, we need to test or evaluate it. The many options for testing can be summarized as follows:

- Try it yourself (or ask someone else to try it)! If you know the languages you are working with and the purpose for which the system was trained, and you have the time, just take a bunch of sentences, translate them and take a look at the resulting translation! With the correct tools, you can also inspect not just the one best translation output by the system, but also the list of *n*-best translations considered by it for each sentence.

- Measure it! There are automatic metrics that one can compute to see how the system behaves (see Rossi & Carré 2022 [this volume]). Most metrics are based on comparing the output of the system to a "reference" translation created by a professional (human) translator (see Rossi & Carré 2022 [this volume]). The meaning of these metrics can vary considerably: some are useful for comparing two different systems with each other, but mean very little by themselves. This applies to *n*-gram or character-based metrics such as BLEU (Papineni et al. 2002), METEOR (Denkowski & Lavie 2014) and CHRF1-3 (Popović 2015). Others are useful to see how much work is needed to turn the automatic translation into the professional translation. This applies to post-editing effort-oriented metrics such as WER (Popović & Ney 2007) or TER (Snover et al. 2006). Yet others tell you about the characteristics of the individual texts translated by the system. Textual metrics include those that measure lexical variety or lexical density. Finally, some metrics are used to rank systems, and will tell you whether your system is preferred to others.

- Get feedback from real-life usage! You can assess whether the system is useful in a real setting, professional or casual, taking into account the purpose of the system. Did you train a system to help people write e-mails? If yes, then ask people to write e-mails using it and tell you about their experience. Did you train a system to help people understand recipes? Then ask users to follow recipes translated by your system and give you feedback.

> Did you train a system to translate legal documents? If so, ask users to use it for their next translation, and report back to you on how they felt about using the system, whether they saved time by using the system, and so on. By gathering this kind of feedback, you will not only be able to judge the current status of your system, but you will also get information that will help you work out how you might improve the system.

Finally, if you have gone to the trouble of creating a custom system, with all the excitement and pain that this might entail, you might want to compare the output of your system with that of a generic system, using any of the relevant testing options above. If your custom system outperforms the generic system, then it is a success. Well done! Otherwise, try to keep having fun!

5 Conclusion

This chapter has provided a brief overview of the customization of MT. Having differentiated between custom MT and generic MT, the chapter stressed the importance of managing expectations when it comes to customization, before introducing the professional roles involved in custom NMT, and asking where MT sits in the translation workflow. Customization through both data and techniques was discussed, and analogies with real-life learning processes were suggested. The chapter concluded with a practical section on tools, customization strategy, data compilation and preparation, training and – finally – testing, in a bid to help readers get hands-on experience of custom MT.

References

Denkowski, Michael & Alon Lavie. 2014. Meteor universal: Language specific translation evaluation for any target language. In *Proceedings of the ninth workshop on Statistical Machine Translation*, 376–380. Baltimore: Association for Computational Linguistics. DOI: 10.3115/v1/W14-3348.

Kenny, Dorothy. 2022. Human and machine translation. In Dorothy Kenny (ed.), *Machine translation for everyone: Empowering users in the age of artificial intelligence*, 23–49. Berlin: Language Science Press. DOI: 10.5281/zenodo.6759976.

Khayrallah, Huda & Philipp Koehn. 2018. On the impact of various types of noise on neural machine translation. In *Proceedings of the 2nd workshop on neural machine translation and generation*, 74–83. Melbourne: Association for Computational Linguistics. https://aclanthology.org/W18-2709.pdf.

Koehn, Philipp. 2020. *Neural Machine Translation.* Cambridge: Cambridge University Press.

Mohiuddin, Tasnim, Philipp Koehn, Vishrav Chaudhary, James Cross, Shruti Bhosale & Shafiq Joty. 2022. *Data selection curriculum for neural machine translation.* DOI: 10.48550/ARXIV.2203.13867.

O'Brien, Sharon. 2022. How to deal with errors in machine translation: Post-editing. In Dorothy Kenny (ed.), *Machine translation for everyone: Empowering users in the age of artificial intelligence*, 105–120. Berlin: Language Science Press. DOI: 10.5281/zenodo.6759982.

Papineni, Kishore, Salim Roukos, Todd Ward & Wei-Jing Zhu. 2002. BLEU: A method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, 311–318. Philadelphia, Pennsylvania, USA: Association for Computational Linguistics. DOI: 10.3115/1073083.1073135.

Pérez-Ortiz, Juan Antonio, Mikel L. Forcada & Felipe Sánchez-Martínez. 2022. How neural machine translation works. In Dorothy Kenny (ed.), *Machine translation for everyone: Empowering users in the age of artificial intelligence*, 141–164. Berlin: Language Science Press. DOI: 10.5281/zenodo.6760020.

Popović, Maja. 2015. Chrf: Character n-gram f-score for automatic MT evaluation. In *Proceedings of the tenth workshop on statistical machine translation*, 392–395. Association for Computational Linguistics. 10.18653/v1/W15-3049.

Popović, Maja & Hermann Ney. 2007. Word error rates. In *Proceedings of the Second Workshop on Statistical Machine Translation* (StatMT '07), 48–55. Prague, Czech Republic. DOI: 10.3115/1626355.1626362.

Rossi, Caroline & Alice Carré. 2022. How to choose a suitable neural machine translation solution: Evaluation of MT quality. In Dorothy Kenny (ed.), *Machine translation for everyone: Empowering users in the age of artificial intelligence*, 51–79. Berlin: Language Science Press. DOI: 10.5281/zenodo.6759978.

Saunders, Danielle. 2021. Domain adaptation and multi-domain adaptation for neural machine translation: A survey. *CoRR* abs/2104.06951. https://arxiv.org/abs/2104.06951.

Snover, Matthew, Bonnie Dorr, Rich Schwartz, Linnea Micciulla & John Makhoul. 2006. A study of translation edit rate with targeted human annotation. In *Proceedings of the 7th conference of the Association for Machine Translation in the Americas: Technical papers*, 223–231. Cambridge, Massachusetts: Association for Machine Translation in the Americas. https://aclanthology.org/2006.amta-papers.25/.