

[AutoML Translation](https://cloud.google.com/translate/automl/) (<https://cloud.google.com/translate/automl/>) [Documentation](#)

# AutoML Translation beginner's guide

## Introduction

Imagine you run a financial reporting service that has an opportunity to expand to new countries. Those markets require that your time-sensitive financial documents are translated in real time. Instead of hiring bilingual finance staff or contracting with a specialist translator, both of which come at a high price due to their domain expertise and your need for quick turnaround, AutoML Translation can help you automate the translation job in a scalable way, allowing you to enter new markets quickly.



## Why is Machine Learning (ML) the right tool for this problem?



Classical programming requires the programmer to specify step-by-step instructions for the computer to follow. But this approach quickly gets unfeasible for translation. Natural language is complex, and translating it is too – rule-based translation stopped being the best approach decades ago. Now machine

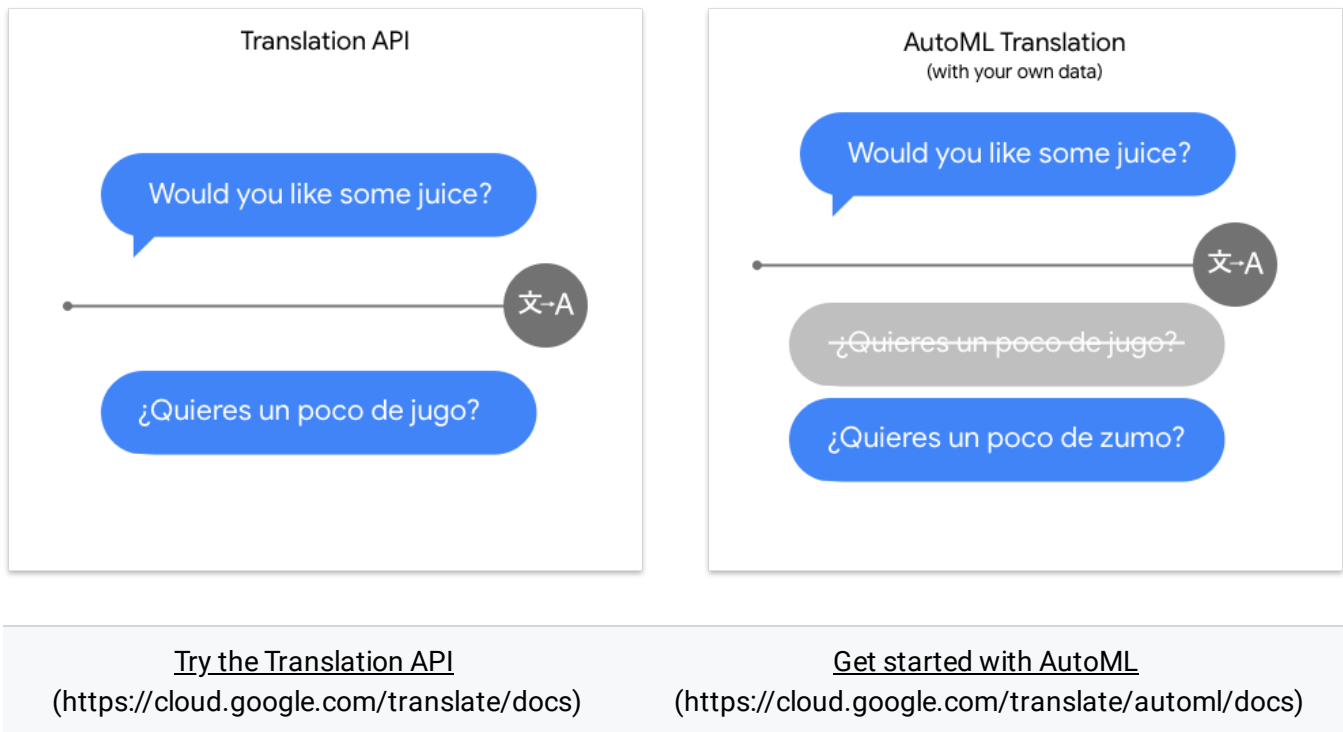
translation is done almost entirely with a statistical approach, with massive parallel corpora taking the place of linguistic experts fine-tuning ever more specialized sets of rules by hand.

You need a system that can generalize to a wide variety of translation scenarios, but is laser-focused on your use case and task-specific linguistic domain in the language pairs you care

about. In a scenario where a sequence of specific rules is bound to expand exponentially, you need a system that can learn from examples. Fortunately, machine learning systems are well-positioned to solve this problem.

## Is the Translation API or AutoML Translation the right tool for me?

The Translation API covers a huge number of language pairs and does a great job with general-purpose text. Where AutoML Translation really shines is for the "last mile" between generic translation tasks and specific, niche vocabularies. Our custom models start from the generic Translation API model, but add a layer that specifically helps the model get the right translation for domain-specific content that matters to you.



## What does machine learning in AutoML Translation involve?




Machine learning involves using data to train algorithms to achieve a desired outcome. The specifics of the algorithm and training methods change based on the problem space. There are many different subcategories of machine learning, all of which solve different problems and work within different constraints. AutoML Translate enables you to perform supervised learning, which involves training a computer to

recognize patterns from translated sentence pairs. Using supervised learning, we can train a custom model to translate domain-specific content you care about.

## Data Preparation

In order to train a custom model with AutoML Translation, you supply matching pairs of sentences in the source and target languages - that is, pairs of sentences that mean the same thing in the language you want to translate from and the one to which you want to translate. Of course, translation isn't an exact science, but the closer in meaning your sentence pairs are, the better your model will work.

### Assess your use case

 While putting together the dataset, always start with the use case. You can begin with the following questions:

- What is the outcome you're trying to achieve?
- What kinds of sentences do you need to translate to achieve this outcome? Is this a task that the Translation API can do out of the box?
- Is it possible for humans to translate these sentences in a way that satisfies you? If the translation task is inherently ambiguous, to the point where a person fluent in both languages would have a hard time doing a satisfactory job, you may find AutoML Translation to be similar in performance.
- What kinds of examples would best reflect the type and range of data your system will need to translate?

A core principle underpinning Google's ML products is human-centered machine learning, an approach that foregrounds responsible AI practices

(<https://ai.google/education/responsible-ai-practices>) including fairness. The goal of fairness in ML is to understand and prevent unjust or prejudicial treatment of people related to race, income, sexual orientation, religion, gender, and other characteristics historically associated with discrimination and marginalization, when and where they manifest in algorithmic systems or algorithmically aided decision-making. You can read more in our guide (<https://cloud.google.com/inclusive-ml>) and find "fair-aware" notes \* in the guidelines below. As you move through the guidelines for putting together your dataset, we encourage you to consider fairness in machine learning where relevant to your use case.

**Fair-aware:** Could your use case or product negatively impact individuals' economic or other important life opportunities? If so, **read more** (<https://cloud.google.com/inclusive-ml/#assess-your-use-case>) about assessing your use case for fairness considerations.

## Source your data

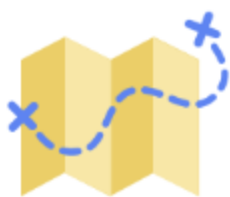


Once you've established what data you will need, you need to find a way to source it. You can begin by taking into account all the data your organization collects. You may find that you're already collecting the data you would need to train a translation model. In case you don't have the data you need, you can obtain it manually or outsource it to a third-party provider.

**Fair-aware:** Review regulations in both your region and the locations your application will serve, as well as existing research or product information in your domain to learn about legal guidelines and common issues.

**Read more** (<https://cloud.google.com/inclusive-ml/#assess-your-use-case>).

## Match data to your problem domain



You're training a custom translation model because you need a model that fits a particular linguistic domain. Make sure your sentence pairs do the best possible job of covering the vocabulary, usage, and grammatical quirks of your industry or area of focus. Find documents that contain typical usages you'd find in the translation tasks you want accomplished, and make sure your parallel phrases match as closely in meaning as you can arrange. Of course, sometimes languages don't map perfectly in vocabulary or syntax, but try to capture the full diversity of semantics you expect to encounter in use if that's possible. You're building on top of a model that already does a pretty good job with general-purpose translation - your examples are the special last step that makes AutoML Translation work for your use case in particular, so make sure they're relevant and representative of usage you expect to see.

## Capture the diversity of your linguistic space

It's tempting to assume that the way people write about a specific domain is uniform enough that a small number of text samples translated by a small number of translators should be sufficient to train a model that works well for anyone else writing about that domain. But we're all individuals, and we each bring our own personality to the words we write. A training dataset



with sentence pairs from a broad selection of authors and translators is more likely to give you a model that's useful for translating writing from a diverse organization. In addition, consider the variety of sentence lengths and structures; a dataset where all the sentences are the same size or share a similar grammatical structure will not give AutoML Translation enough information to build a good model that captures all the possibilities.

## Keep humans in the loop



If it's at all feasible, make sure a person who understands both languages well has validated the sentence pairs match up correctly and represent understandable, accurate translations. A mistake as simple as misaligning the rows of your training data spreadsheet can yield translations that sound like nonsense. High-quality data is the most important thing you can provide to AutoML Translation to get a model that's usable for your business.

## Clean up messy data



It's easy to make mistakes when preprocessing data, and some of those mistakes can really confuse an AutoML Translation model. In particular, look for these easy-to-miss data issues:

- Identical source and target sentences
- Misaligned source and target sentences
- Sentences that don't match the specified language; for example, English sentences in a Chinese dataset
- Sentences with mixed language; for example a target sentence accidentally containing untranslated words from the source sentence
- Sentences with typographical or grammatical errors; your model is likely to learn these errors.
- Duplicate sentences in the training and test sets ([Learn more](#) (#training-set) about train and test sets)
- Multiple sentences in the same text pair. Training on a dataset where many items have more than about 50 tokens (words) in them yields lower quality models. Split items into

individual sentences where possible.

### How AutoML preprocesses your data

AutoML Translation will stop parsing your data input file when:

- There is invalid formatting
- There is an unreasonably long sentence pair (10 MB)
- The file uses an encoding other than UTF-8

AutoML Translation ignores errors for problems it can detect, such as:

- A <tu> element in a TMX file doesn't have the source language or target language.
- One of the input sentence pairs is empty.

In AutoSplit mode, AutoML Translation does additional processing:

- After the dataset is uploaded, it removes sentence pairs with identical source sentences.
- It randomly splits your data into three sets with split ratio 8:1:1 before training.

### Consider how AutoML Translation uses your dataset in creating a custom model



Your dataset contains training, validation and testing sets. If you do not specify the splits (see [Preparing your training data](https://cloud.google.com/translate/automl/docs/prepare) (<https://cloud.google.com/translate/automl/docs/prepare>) and your dataset contains under 100,000 sentence pairs, then AutoML Translation automatically uses 80% of your content documents for training, 10% for validating, and 10% for testing. If your data is larger than that, you'll need to perform your own data split.

### Training Set



the neural network.

The vast majority of your data should be in the training set. This is the data your model "sees" during training: it's used to learn the parameters of the model, namely the weights of the connections between nodes of

## Validation Set



The validation set, sometimes also called the "dev" set, is also used during the training process. During model learning, the framework uses the training set to train a suite of candidate models, and then uses the model's performance on the validation set to choose the best model generated. It uses the model's performance on the validation set to tune the model's hyperparameters, which are variables that specify the model's structure. If you used the training set to tune the hyperparameters, the model would end up overly focused on your training data. Using a somewhat novel dataset to fine-tune model structure means your model will generalize better.

## Test Set



The test set is not involved in the training process at all. Once the model has completed its training entirely, we use the test set as an entirely new challenge for your model. The performance of your model on the test set is intended to give you a pretty good idea of how your model will perform on real-world data.

## Manual Splitting



AutoML can split your data into training, validation, and test sets for you, or you can do it yourself if you want to exercise more control over the process, if you'd prefer a different percentage split, or if there are specific examples that you're sure you want included in a certain part of your model training lifecycle.

**Fair-aware:** If you have scarce data about a particular subgroup, make sure that data is spread representatively between your training and test sets by performing the train/test split yourself. **Read more** (<https://cloud.google.com/inclusive-ml/#data-guidelines>).

## Prepare your data for import



Once you've decided if a manual or automatic split of your data is right for you, there are two ways to add data in AutoML Translation:

- You can import data as a tab-separated values (TSV) file containing source and target sentences, one sentence pair per line.
- You can import data as a TMX file, a standard format for providing sentence pairs to automatic translation model tools ([learn more about the supported TMX format](https://cloud.google.com/translate/automl/docs/prepare) (<https://cloud.google.com/translate/automl/docs/prepare>)). If the TMX file contains invalid XML tags, AutoML Translation ignores them. If the TMX file does not conform to proper XML and TMX format – for example, if it is missing an end tag or a <tmx> element – AutoML Translation will not process it. AutoML Translation also ends processing and returns an error if it skips more than 1024 invalid <tu> elements.

## Evaluate

Once your model is trained, you will receive a summary of your model performance. Click the **Train** tab after the model has completed training to view a detailed analysis.

What should I keep in mind before evaluating my model?



Debugging a model is more about debugging the data than the model itself. If your model starts acting in an unexpected manner as you're evaluating its performance before and after pushing to production, you should return and check your data to see where it might be improved.

## BLEU score

The BLEU score is a standard way to measure the quality of a machine translation system. AutoML Translation uses a BLEU score calculated on the test data you've provided as its primary evaluation metric. ([Learn more about BLEU scores](https://cloud.google.com/translate/automl/docs/evaluating#understanding_the_bleu_score) ([https://cloud.google.com/translate/automl/docs/evaluating#understanding\\_the\\_bleu\\_score](https://cloud.google.com/translate/automl/docs/evaluating#understanding_the_bleu_score))).

The Google NMT model, which powers the Translation API, is built for general usage. It may not be the best solution for you if you are looking for specialized translation in your own fields. The



trained custom model usually works better than the NMT model in the fields that your training set is related to.

After you train the custom model with your own dataset, the BLEU score of the custom model and Google NMT model will be shown in the **Train** tab. There is also a BLEU score performance gain from the custom model on the **Train** tab. The higher the BLEU score, the better translations your model can give you for sentences that are similar to your training data. If the BLEU score falls in the range 30-40, the model is considered to be able to provide good translations.

## Testing your model



Even if the BLEU score looks okay, it's a good practice to sanity check the model yourself to make sure its performance matches your expectations. If your training and test data are drawn from the same incorrect set of samples, the scores might be excellent even if the translation is nonsense! Come up with some sanity-check examples to input on the AutoML Translation **Predict** tab and compare with the results from the Google NMT base model, or use the instructions on that tab to call the AutoML API to use your model in automated tests. You might notice that your model comes up with the same predictions as the base model, especially on short sentences or if you have a smaller training set. This isn't unexpected - the base model is already pretty good for a wide variety of use cases. Try some longer or more complex sentences. However, if all of your sentences come back identical to the predictions from the base model, this may indicate a data problem.

**Fair-aware:** Think carefully about your problem domain and its potential for unfairness and bias. Come up with cases that would adversely impact your users if they were found in production, and test those first.

**Read more** (<https://cloud.google.com/inclusive-ml/#predict>).

If there's a mistake that you're particularly worried about your model making (for example, a confusing feature of your language pair that often trips up human translators, or a translation mistake that might be especially costly in money or reputation) make sure your test set or procedure covers that case adequately for you to feel safe using your model in everyday tasks.

**Fair-aware:** If you have a use case that warrants fairness considerations, read more about how to use your model in a manner that mitigates biases or adverse outcomes. **Read more**

(<https://cloud.google.com/inclusive-ml/#use-your-model>).

*Except as otherwise noted, the content of this page is licensed under the [Creative Commons Attribution 4.0 License](https://creativecommons.org/licenses/by/4.0/) (https://creativecommons.org/licenses/by/4.0/), and code samples are licensed under the [Apache 2.0 License](https://www.apache.org/licenses/LICENSE-2.0) (https://www.apache.org/licenses/LICENSE-2.0). For details, see our [Site Policies](https://developers.google.com/terms/site-policies) (https://developers.google.com/terms/site-policies). Java is a registered trademark of Oracle and/or its affiliates.*

*Last updated October 15, 2019.*