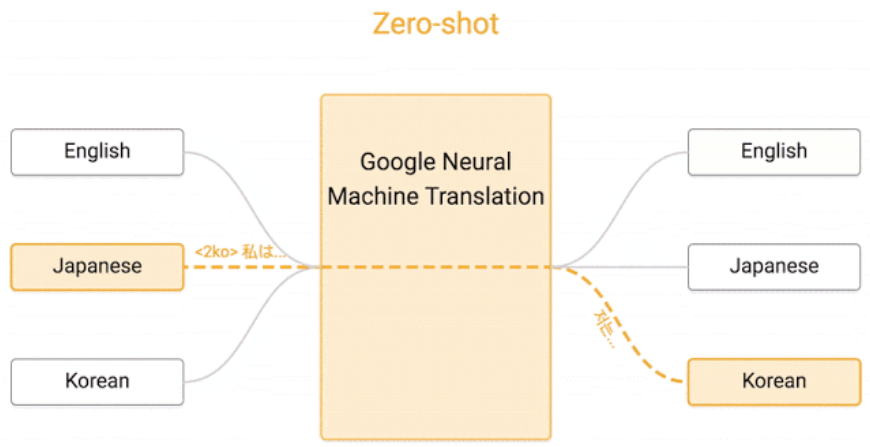
Daniil Korbut Follow
Junior Data Scientist at Statsbot
Aug 1 · 6 min read

Machine Learning Translation and the Google Translate Algorithm

The basic principles of machine translation engines



Google Machine Translation

Every day we use different technologies without even knowing how exactly they work. In fact, it's not very easy to understand engines powered by machine learning. The Statsbot team wants to make machine learning clear by telling data stories in this blog. Today, we've decided to explore machine translators and explain how the Google Translate algorithm works.

. . .

Years ago, it was very time consuming to translate the text from an unknown language. Using simple vocabularies with word-for-word translation was hard for two reasons: 1) the reader had to know the grammar rules and 2) needed to keep in mind all language versions while translating the whole sentence.

Now, we don't need to struggle so much– we can translate phrases, sentences, and even large texts just by putting them in Google

Translate. But most people don't actually care how the engine of machine learning translation works. This post is for those who do care.

Deep learning translation problems

If the Google Translate engine tried to kept the translations for even short sentences, it wouldn't work because of the huge number of possible variations. The best idea can be to teach the computer sets of grammar rules and translate the sentences according to them. If only it were as easy as it sounds.

If you have ever tried learning a foreign language, you know that there are always a lot of exceptions to rules. When we try to capture all these rules, exceptions and exceptions to the exceptions in the program, the quality of translation breaks down.

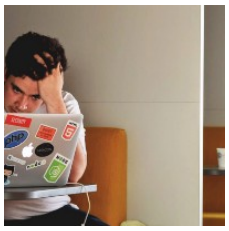
> *Modern machine translation systems use a different approach: they allocate the rules from text by analyzing a huge set of documents.*

Creating your own simple machine translator would be **a great project for any data science resume**.

Data Scientist Resume Projects

Machine learning problems set to build a data scientist CV without work experience

blog.statsbot.co

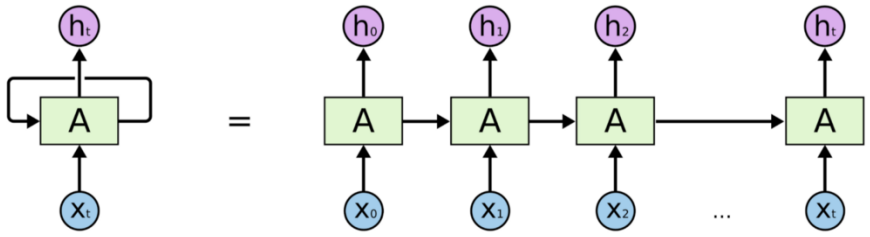


Let's try to investigate what hides in the "black boxes" that we call machine translators. Deep neural networks can achieve excellent results in very complicated tasks (speech/visual object recognition), but despite their flexibility, they can be applied only for tasks where the input and target have fixed dimensionality.

Recurrent Neural Networks

Here is where Long Short-Term Memory networks (LSTMs) come into play, helping us to work with sequences whose length we can't know a priori.

LSTMs are a special kind of recurrent neural network (RNN), capable of learning long-term dependencies. All RNNs look like a chain of repeating modules.



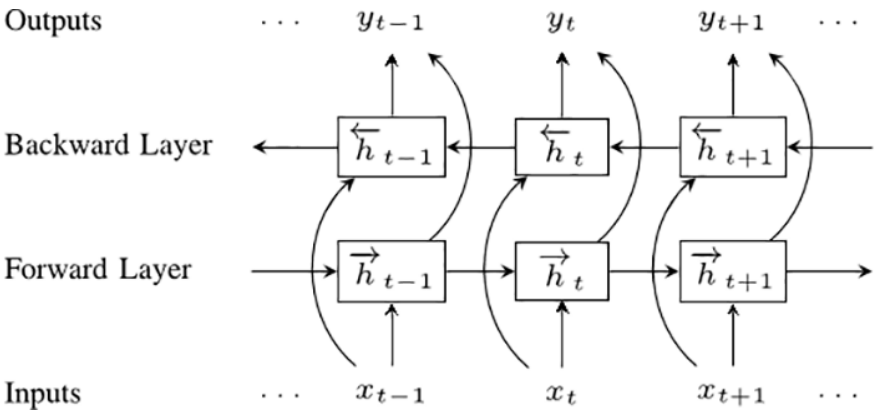
An unrolled recurrent neural network.

Unrolled recurrent neural network

So the LSTM transmits data from module to module and, for example, for generating Ht we use not only Xt, but all previous input values X. To learn more about structure and mathematical models of LSTM, you can read the great article "Understanding LSTM Networks."

Bidirectional RNNs

Our next step is bidirectional recurrent neural networks (BRNNs). What a BRNN does, is split the neurons of a regular RNN into two directions. One direction is for positive time, or forward states. The other direction is for negative time, or backward states. The output of these two states are not connected to inputs of the opposite direction states.

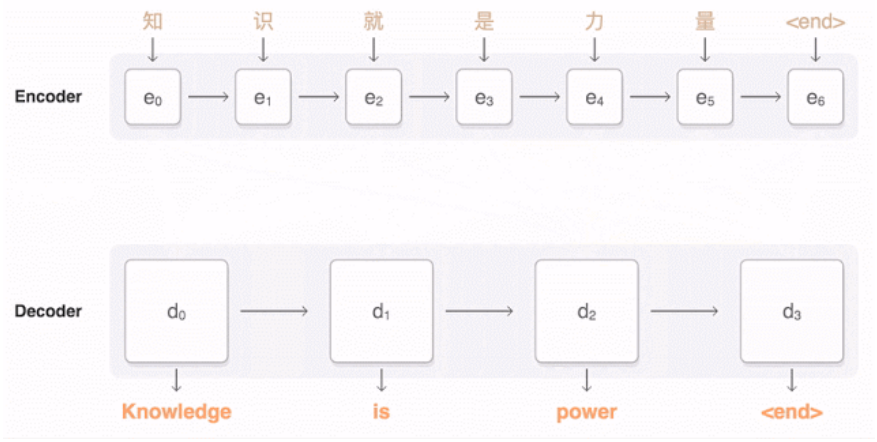


Bidirectional recurrent neural networks

To understand why BRNNs can work better than a simple RNN, imagine that we have a sentence of 9 words and we want to predict the 5th word. We can make it know either only the first 4 words, or the first 4 words and last 4 words. Of course, the quality in the second case would be better.

Sequence to sequence

Now we're ready to move to sequence to sequence models (also called seq2seq). The basic seq2seq model consist of two RNNs: an encoder network that processes the input and a decoder network that generates the output.



Sequence to sequence model

Finally, we can make our first machine translator!

However, let's think about one trick. Google Translate currently supports 103 languages, so we should have 103x102 different models for each pair of languages. Of course, the quality of these models varies according to the popularity of languages and the amount of documents needed for training this network. The best that we can do is to make one NN to take any language as input and translate into any language.

Google Translate

That very idea was realized by Google engineers at the end of 2016. Architecture of NN was build on the seq2seq model, which we have already studied.

The only exception is that between the encoder and decoder there are 8 layers of LSTM-RNN that have residual connections between layers with some tweaks for accuracy and speed. If you want to go deeper with that, take a look at the article Google's Neural Machine Translation System.

> *The main thing about this approach is that now the Google Translate algorithm uses only one system instead of a huge set for every pair of languages.*

The system requires a "token" at the beginning of the input sentence which specifies the language you're trying to translate the phrase into.

This improves translation quality and enables translations even between two languages which the system hasn't seen yet, a method termed "Zero-Shot Translation."

What means better translation?

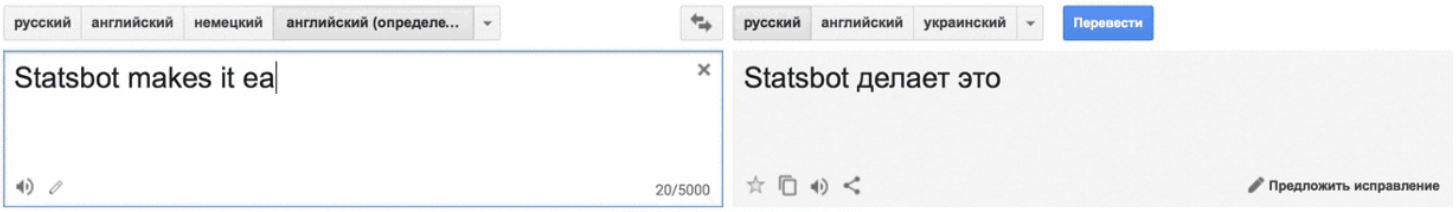
When we're talking about improvements and better results from Google Translate algorithms, how can we correctly evaluate that the first candidate for translation is better than the second?

It's not a trivial problem, because for some commonly used sentences we have the sets of reference translations from the professional translators, that have, of course, some differences.

There are a lot of approaches that partly solve this problem, but the most popular and effective metric is BLEU (bilingual evaluation understudy). Imagine, we have two candidates from machine translators:

> *Candidate 1: Statsbot makes it easy for companies to closely monitor data from various analytical platforms via natural language.*

> *Candidate 2: Statsbot uses natural language to accurately analyze businesses' metrics from different analytical platforms.*



Although they have the same meaning they differ in quality and have different structure.

Let's look at two human translations:

> *Reference 1: Statsbot helps companies closely monitor their data from different analytical platforms via natural language.*

> *Reference 2: Statsbot allows companies to carefully monitor data from various analytics platforms by using natural language.*

Obviously, Candidate 1 is better, sharing more words and phrases compared to Candidate 2. This is a key idea of the simple BLEU approach. We can compare n-grams of the candidate with n-grams of the reference translation and count the number of matches (independent from their position). We use only n-gram precisions, because calculating recall is difficult with multiple refs and the result is the geometric average of n-gram scores.

. . .

Now you can evaluate the complex engine of machine learning translation. Next time when you translate something with Google Translate, imagine how many millions of documents it analyzed before giving you the best language version.

YOU'D ALSO LIKE:

Recommendation System Algorithms

Main existing recommendation engines and how they work

blog.statsbot.co

Text Classifier Algorithms in Machine Learning

Key text classification algorithms with use cases and tutorials

blog.statsbot.co

Time Series Anomaly Detection Algorithms

The current state of anomaly detection techniques in plain language

blog.statsbot.co