LaTeX Author Guidelines for CVPR Proceedings

Yuan An

In the last article, we knew something about face recognition using the neural network. Today, I continue to read the article of Geitgey [2]. The part four of this article is talking about another application of the neural network– Machine Translation. A good example of machine translation is Google Translate, which can instantly translate between 100 different human languages in a magic way. And it is even available on mobile phones and smart-watches. It does bring convenience to us even in academic scene.

Over the past years, deep learning has totally changed approach to machine translation. Deep learning researchers who know almost nothing about language translation are beating the best expert-built language translation system in the world using simple machine learning solutions.

The breakthrough technology is called sequence-to-sequence learning. It is very powerful technique that can be used to solve many kinds problems.

The simplest approach to make computer translating human language is to replace every word in a sentence with the translate word in the target language. As shown in Fig. 1, this is a simple example of translating from Spanish to English word-by-word.

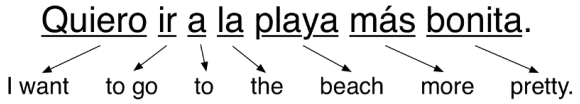

Figure 1. Replace each Spanish word with the matching English word.

It is easy to implement but the results are bad for its ignoring grammar and context. So the next thing that we should do is to add language-specific rules to improve the results. *E.g*. we might translate common two-word phrases as a single group, and swap the order nouns and adjectives since they usually appear in reverse order in Spanish from how they appear in English (see Fig. 2).

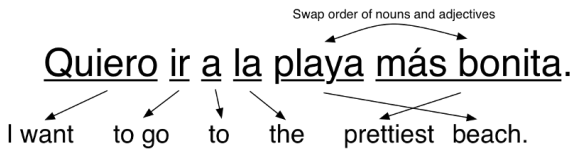

Figure 2. Swap the order of nouns and adjectives.

It did work better. So theoretically, if more rules is keeping being added until it can handle every part of grammar, the program should be able to translate any sentence. This is how earliest machine translation systems worked. Linguists came up with complicated rules and programmed them in one-by-one. Some of the smartest linguists in the world labored for years during the Cold War to create translation systems as a way wo interpret Russian communications more easily (more thing about this is on Wikipedia: *Georgetown–IBM experiment*).

But this only worked for simple, plainly-structed documents like weather reports. It didn't work reliably for real-world documents. But the human language doesn't follow a fixed set of rules. Human languages are full of special case, regional variations or just flat out rule-breaking. The way we speak English more influenced by who invaded wo hundreds of years ago than it is by someones sitting down and defining grammar rules (called Middle English).

So a statistic method should be used to make computers translate better. The new translation approaches were developed using models based on probability and statistics instead of grammar rules.

Building a statistics-based translation system requires lots of training where the exact same text is translated into at least two languages. This double-translated text is called *pararel corpora*. And computers can use parallel corpora to guess how to convert text from one language to another.

The most important thing for computers is thinking in probabilities. The fundamental difference with statistical translation system is that they don't try to generate one exact translation instead they generate thousands of possible translations and then rank those translations by likely each is to be correct. They estimate how "correct" something is by how similar it is to the training data.

The first step is break original sentence into chunks (as shown in Fig. 3) that can each be easily translated.

Figure 3. Break original sentence into chunks.

Then find all possible translations for each chunk how humans have translated those same chunks of words in training data. We are looking how actual people translated these same chunks of words instead of looking up these chunks in

a simple translation dictionary. It will help us capture all of the different ways they can be used in different contexts (see Fig. 4).

Figure 4. How actual people translate chunks.

Some of these possible translations are used more frequently than others. So we can give it a score based on how frequently each translation appears in the training data.

The third step is generating all possible sentences and find the most likely one. That is to say that use every possible combination of these chunks to generate a bunch of possible sentences. Then need to scan through all of these generated sentences to find the one that is that sounds the "most human". We can give the sentence that would not be very similar to any sentences a low probability score while give the sentence similar to those in dataset a high probability score. After doing this, we can pick out the sentence that has the most likely chunk translations while also being the most similar overall to real English sentences.

But it also have limitations. It is complicated to build and maintain. Every new pair of languages you want to translate requires experts to tweak and tune a new multi-step translation pipeline.

However, there is a approach to make computer translate better without all those expensive experts. The core of machine translation is a black box system that learns how to translate by itself.

In 2014, KyunHyun Cho's team made a breakthrough [1]. They found a way to apply deep learning to build this black box system. Their deep learning model takes in a parallel corpora and uses it to learn how to translate between those two languages without any human intervention.

Two big ideas make this possible– recurrent neural networks and encodings. By combining these two ideas in a clever way, a self-learning translation system can be built. These two ideas have been introduced in previous articles.

After using an RNN to encode a sentence into a set of unique numbers, take two RNNs and hook them up end-to-end (see Fig. 5

The first RNN could generate the encoding that represents a sentence. Then the second RNN could generate the encoding and just do the same logic in reverse to decode the original sentence again. Of course it is not useful to encode and decode the original sentence. But we can train

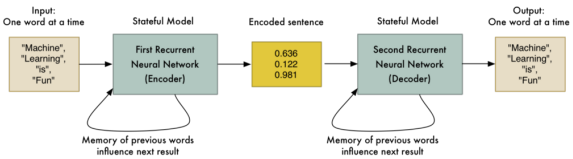

Figure 5. Hook up two RNNs end-to-end.

the second RNN to decode the sentence into Spanish instead of English (as shown in Fig. 6).

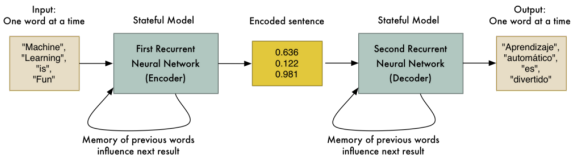

Figure 6. Replace the second RNN with Spanish.

Just like that, there is a generic way of converting a sequence of English words into an equivalent sequence of Spanish words.

This approach works for almost any kind of sequence-to-sequence problems. And the power of sequence-to-sequence models is infinity.

A team of Google replaced the first RNN with a CNN. This allows the input to be a picture instead of a sentence (the structure of the model is shown in Fig. 7).

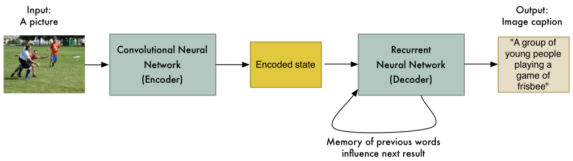

Figure 7. The model structure of the Google's team.

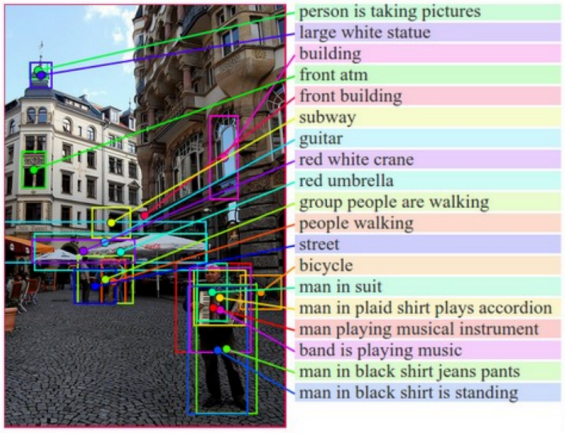

Figure 8. Image from Andrej Karpathy's paper.

Andrej Karpathy expanded on these ideas to build a system capable of describing images in great detail by processing multiple regions of an image separately (see Fig. 8) [3].

References

[1] K. Cho, B. van Merrienboer, Ç. Gülçehre, F. Bougares, H. Schwenk, and Y. Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

[2] A. Geitgey. Machine learning is fun. `https://medium.com/@ageitgey/machine-learning-is-fun-80ea3ec3c471`.

[3] A. Karpathy and F. F. Li. Deep visual-semantic alignments for generating image descriptions. In *CVPR*, pages 3128–3137, 2015.