Neural machine translation
The basic concepts, principle of operation and review of approaches

Neural machine translation (NMT) is an approach to machine translation that uses an artificial neural network to predict the likelihood of a sequence of words, typically modeling entire sentences in a single integrated model. [1]

The main advantage of this approach is that a single system can be trained directly from the source and target text, no longer requiring a set of specialized systems used in statistical machine learning.

Unlike a traditional phrase-based translation system, which consists of many small subcomponents that are configured separately, neural machine translation attempts to build and train one large neural network that reads a sentence and produces the correct translation.

The goal of the NMT is to create a fully trainable model, each component of which is customized based on training corpuses to maximize translation quality.

Thus, neural machine translation systems are said to be complex systems since only one model is required for translation.

The strength of the NMT lies in its ability to learn directly end-to-end conversion of input text to related output text. [2]

Deep learning applications appeared first in speech recognition in the 1990s. The first scientific paper on using neural networks in machine translation appeared in 2014, followed by a lot of advances in the following few years. (Large-vocabulary NMT, application to Image captioning, Subword-NMT, Multilingual NMT, Multi-Source NMT, Character-dec NMT, Zero-Resource NMT, Google, Fully Character-NMT, Zero-Shot NMT in 2017) In 2015 there was the first appearance of a NMT system in a public machine translation competition (OpenMT'15). WMT'15 also for the first time had a NMT contender; the following year it already had 90% of NMT systems among its winners. [1]

What is the essence of neural machine translation?

Such systems also train on large numbers of pairs of sentences, but word matching and breakdown into phrases are no longer needed. The system simply tries to first encode the original sentence into an abstract set of numbers - and then decode words from the numbers, but in a different language. Having “read” the sentence to the end, and simultaneously both from left to right and from right to left, the neural network begins to decode and predict the words of the translation, and each predicted word is used to predict the next word. Thus, for a more accurate choice of words for translation, the context of the entire original sentence is used, as well as the context of all previous predicted words. This is the main difference between neural systems and systems based on individual phrases, in which the context is limited by the previous 4-7 words. And it's more like how a person translates.

In the process of training neural systems, there is a constant comparison of each predicted word (and therefore the most probable according to the state of the model at the moment) with the “correct” word, that is, with the word that is in this place of the target training sentence. If the predicted word does not coincide with it, then the so-called "update" of the model parameters (redistribution of weights in the neural network) occurs, which increases the probability of the correct word and reduces the probability of the words that were determined as the most probable at this training step. This is repeated very, very many times. As a result, the neural system goes through all the millions of pairs of sentences, which are necessarily mixed in random order, 3-4 times. The algorithm stops when there is no further improvement in word prediction.

Neural translation systems differ mainly in their parameters, namely in how many numbers (more strictly speaking, in which vectors and matrices) words are encoded at different stages, what are the sizes of these vectors and matrices, what mathematical operations they are connected to each other ("neural" connections, as in the human brain, are here to some extent modeled using linear algebra, hence the name) and how memorization of important words and “forgetting” of less important ones is ensured.

All modern systems are equipped with the so-called "attention mechanism", which, when predicting the next word, seems to focus on one or more words of the original sentence, adding this information with the encoded full context. Here, again, there is a similarity with the behavior of a human translator, who also usually reads the entire sentence first, and then, in the process of translation, looks at individual source words and phrases, either already translated or those that have yet to be translated.

The attention mechanism is somewhat similar to word alignment (that is, the search for word-by-word matches in texts in different languages), but this attention is "soft" - it is not always possible to say exactly which word was translated by the system at a certain point in time, you can just to see which words influenced the prediction of this particular word in translation more than others. Systems with the latest generation architecture, called Transformer, have several levels of such "attention", and not only attention to the original proposal, but also to the already partially predicted translation, the so-called self-attention.

In order to create a neural machine translation system from scratch (as well as for statistical translation systems), you need, first of all, several million pairs of sentences, where the translation of the original sentence was made by a human. For languages with a large number of speakers, such data is easy to find even in the public domain.

Secondly, you need a program to train the neural network. There are also quite a few such free code programs, but most of them are based on data types and neural algorithms from TensorFlow or PyTorch. Almost all programs are written in Python. They allow you to set neural network parameters, training parameters, etc. Determining the appropriate parameters requires a good knowledge of machine learning, probability and statistics, as well as linear algebra and mathematics in general.

Knowledge of computational linguistics is needed in order to choose the form of representation of words and sentences in a neural network well. Each word is encoded into a vector of real numbers measuring 600-1000. In the process of training, the system itself learns that certain dimensions have their own internal semantics. For example, vectors for the words "king" and "kings" can be similar, that is, the distance between them in the multidimensional space of such vectors will not be very large.

All these semantic connections allow neural systems to efficiently memorize and produce translations using synonyms and, in general, words that they have not seen anywhere during training in the "human" translation of words from a given sentence. This is another significant difference from statistical translation systems based only on "learned" words and phrases. However, the system can learn well these semantic connections only when it has seen these words quite often during training. Therefore, neural systems often make mistakes in translating rare words - the vectors calculated for them are simply not reliable enough.

To reduce the number of rare words, as well as to be able (or try) to translate completely unfamiliar words, algorithms for dividing words into parts are used, so that each part occurs quite often in the corpus. Thus, the system learns to translate whole frequent words (for example, 10 thousand most frequent words of the language) and fragments of less frequent words. It also helps make the system faster. When predicting each next word in the translation, the system must always simply calculate the probability for each word of the target language from its vocabulary in order to select the most likely word. When using fragments of words, this probability is calculated 20 or 50 thousand times, and not 200 or 300 thousand, if we are talking about all words of the language.

To train a neural translation system, you need a powerful computer with one or more latest generation graphics cards specially designed for fast operations with vectors and matrices. The training of the system lasts from one day to a week or more, depending on the number of training pairs of sentences and network parameters. To use an already trained neural translation system, a graphic card is also recommended, since then the translation speed will increase from 10-30 words per second to 300 or more words per second.

While translation of neural systems can be of fairly good quality, there are still many challenges to be solved by scientists and developers.

First, as already mentioned, the problem of translating rare words remains even when translating a word in parts. Moreover, such a translation sometimes leads to the fact that the system begins to invent new words, and also often distorts the names of people and names that it did not encounter in the learning process.

Secondly, sometimes a good structure of the translation, the fluency of the language can "lull" the reader or the translation editor so that a serious mistake in translation will go unnoticed. As a result, even very important words may remain untranslated due to the imperfection of the attention mechanism, or be translated incorrectly only because such a translation, in combination with another word, forms a frequently encountered phrase in the target language.

In addition, on very “heavy”, unusual texts, the system can generally “get confused”, trying to produce something that sounds good in the target language, but has little to do with what was in the original sentence (something like “the decision of the decision ", with repeated words.)

The main problem of spreading neural translation systems to a larger number of language pairs is the lack of parallel corpora for teaching rare language pairs. Moreover, they are rare not always because of the number of carriers (there can be tens and hundreds of millions of them), but because there are very few documents in these languages and their translations in written and especially digitized form.

Finally, another problem that, however, can be solved is the dependence of neural networks on the type of data on which they train. So, if you train neural translation only on long pairs of sentences (from 20 words each), then the system will be unable to translate a short sentence or even a single word. In my practice, there was a case, for example, that the interjection "eh" was translated by the system as "I don't know what to say." Therefore, it is imperative to randomly "slip" the system and words from the dictionary with translation, and individual phrases; and numbers (phone numbers, for example), and names, and smilies from brackets so that she learns to copy them without loss. Short and long sentences, with or without a dot at the end. Try now to put or not to put a full stop at the end of a sentence in Google Translate and, as they say, feel the difference. As practice shows, all this has little effect on the overall quality of the translation (often determined automatically using metrics such as BLEU, which compare the translation of the system with a professional translation), but for the end users all these punctuation marks are in the right place and in the right form, correct translations of names, etc. are very important. [3]

Some background work which is common to almost all the neural machine translation systems

1. Recurrent Language Model

A *recurrent neural network (RNN)* is a neural network that consists of a hidden state h and an optional output y which operates on a variable length sequence $x = (x_1 ... x_T)$. At each time step t, the hidden state h_t of the RNN is updated by:

$$h_t = f(h_{t-1}, x_t) \tag{1}$$

where the f is non-linear activation function which is usually implemented with LSTM cell. Using softmax function with vocabulary size V, an RNN can be trained to predict the distribution over x_t given the history of words $(x_{t-1}, x_{t-2}, ...x_1)$ at each time step t. By combining the probabilities at each time step, we can compute the probability of the sequence x (eg: target language sentence) using

$$p(x) = \prod_{t=1}^{T} p(x_t | x_{t-1}, \dots, x_1) \tag{2}$$

which is called the Recurrent Language Model (RLM). [4]

2. Encoder-Decoder Architecture

Though RNN Encoder-Decoder architecture was proposed for a machine translation task, it remains the base model for most of the NLP sequence-to-sequence models (and especially machine translation). [4]

An encoder-decoder neural model (figure 1), from a probabilistic perspective, is a general method to learn the conditional distribution over variable length sequence given yet another variable length sequence, e.g. $p(y_1, \ldots, y_{T'} | x_1, \ldots, x_T)$. An encoder is an RNN which reads each symbol in input sequence *(x)* one word at a time till it encounters end-of-sequence symbol. The hidden state of the RNN at the last time step is the summary c of the whole input sequence. The decoder operates very similar to RLM discussed previously except that the hidden state of the decoder h_t now depends on the summary c too. Hence the hidden state of the decoder at time step t is calculated by

$$h_t = f(h_{t-1}, y_{t-1}, c) \tag{3}$$

and the conditional distribution of the next symbol (for e.g. next word in target language sentence given source language sentence) is

$$p(y_t | y_1, \ldots, y_{t-1}) = g(h_t, y_{t-1}, c) \tag{4}$$

The two components of the encoder-decoder model are jointly trained to maximize the conditional log-likelihood

$$\frac{1}{N} \sum_{n=1}^{N} log p_\theta(y_n | x_n) \tag{5}$$

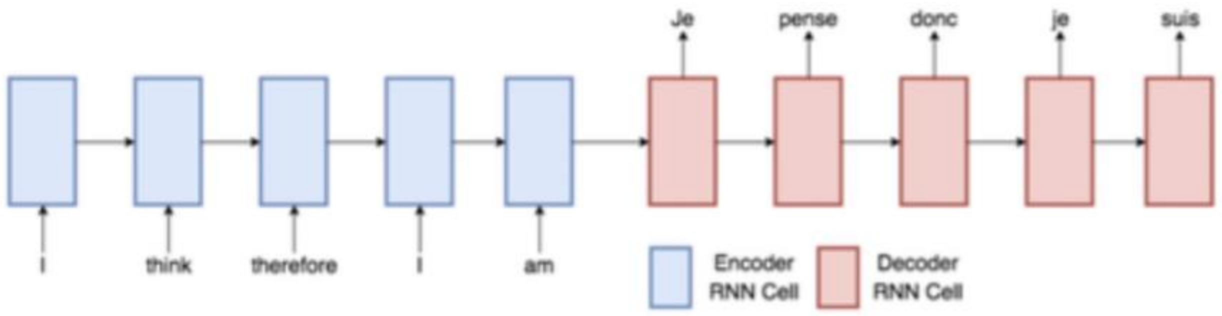

Figure 1. An encoder-decoder architecture [4]

NMT Methods

Motivated from success of deep neural networks and their ability to represent a linguistic unit with a continuous representation, Kalchbrenner et al. [5] propose a class of probabilistic translation models, Recurrent Continuous Translation Model (RCTM) for machine translation. The RCTM model has a generation aspect and a conditional aspect. The generation of a sentence in target language is modelled with target Recurrent language model. The conditioning on the source sentence is modelled with a Convolutional Neural Network (CNN). In their model, CNN takes a sentence as input and generates a fixed size representation of this source sentence. This representation of source sentence is presented to the Recurrent Language Model to produce the translation in target language. The entire model (CNN and RNN) is trained jointly with back-propagation.

To the best of our knowledge, this is the first work which explores the idea of modelling the task of machine translation entirely with neural networks, with no component from statistical machine translation systems. They propose two CNN architectures to map source sentence into fixed size continuous representation.

The Convolutional Sentence Model (CSM) creates a representation for a sentence that is progressively built up from representations of the n-grams in the sentence. The CSM architecture embodies a hierarchical structure, similar to parse trees, to create a sentence representation. The lower layers in the CNN architecture operate locally on n-grams and the upper layers act increasingly globally on the entire sentence. The lack of need of parse tree makes it easy to apply these models to languages for which parsers are not available. Also, generation of the sentence in target language is not dependent on one particular parse tree. Similar to CSM, the authors propose another CNN model called Convolutional n-gram model (CGM). The CGM is obtained by truncating the CSM at the level where n-grams are represented for the chosen value of n. The CGM can also be inverted (icgm) to obtain a representation for a sentence from the representation of its n-grams. The transformation icgm unfolds the n-gram representation onto a representation of a target sentence with m target words (where m is also predicted by the network according to the Poisson distribution). The pictorial representation of two models is shown in figure 2.

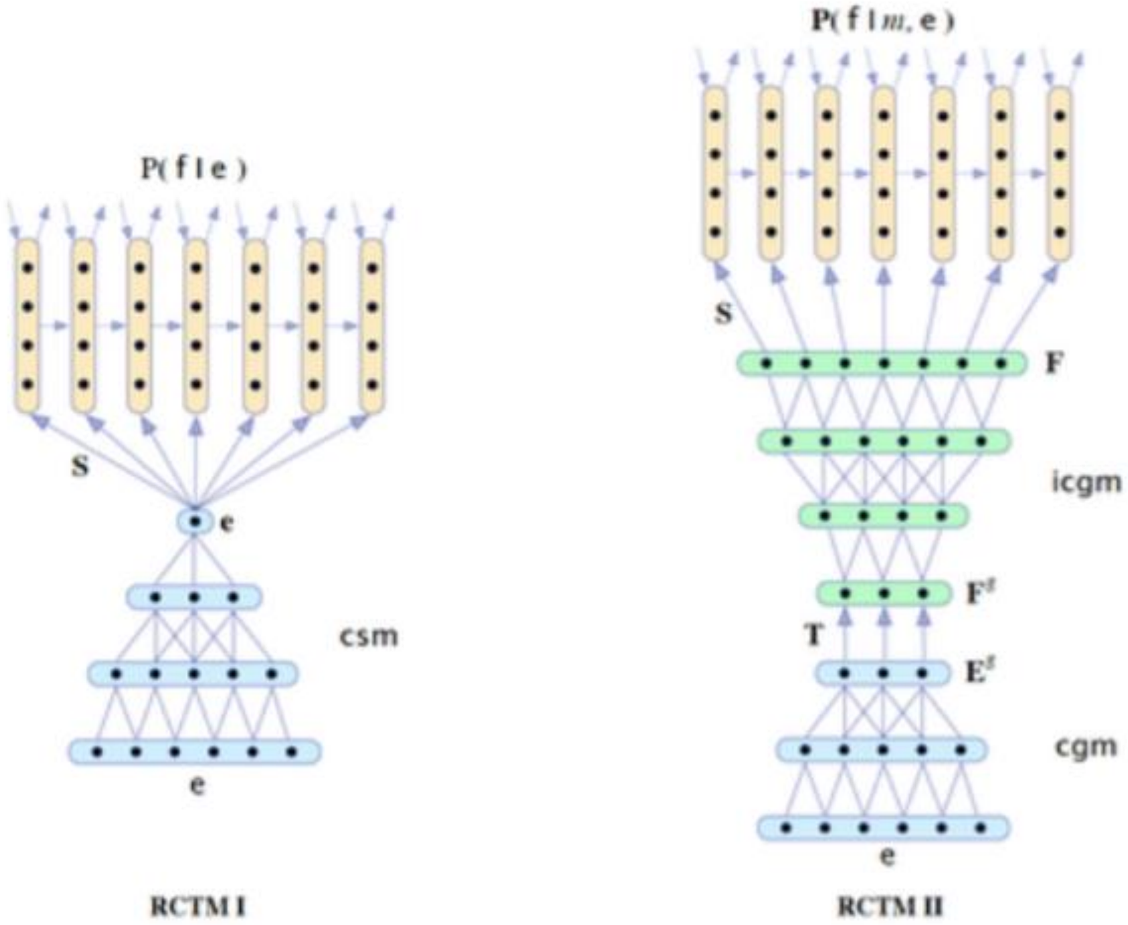

Figure 2. A graphical depiction of the two RCTMs. Arrows represent full matrix transformations while lines are vector transformations corresponding to columns of weight matrices.

The experimentation is performed on a bilingual corpus of 144953 pairs of sentences less than 80 words in length from the news commentary section of the Eighth Workshop on Machine Translation (WMT) 2013 training data. The source language is English and the target language is French. A low perplexity value achieved by RCTMs on test set as compared to IBM models (model 1-4) suggests that continuous representations and the transformations between them make up well for the lack of explicit alignments. To make sure that RCTM architecture (with CGM) doesn't just take bag-of-words approach, they change the ordering of the words in the source sentence and train their model. This model achieves much lower perplexity values which proves that the model is indeed sensitive to source sentence structure. They also compare the performance of the RCTM model with cdec system. cdec employs 12 engineered features including, among others, 5 translation models, 2 language model features and a word penalty feature (WP). RCTM models achieve comparable performance (marginally better) than the cded system on BLEU score. The results indicate that the RCTMs are able to learn both translation and language modelling distributions without explicitly modelling them.

Cho et al. [6] propose a RNN encoder-decoder architecture very similar to the one proposed above but with one major difference. While Kalchbrenner et al. [5] use CNN to map a source sentence into a fixed-sized continuous representation, Cho et al. [6] use an encoder RNN to map source sequence into a vector. However, they use this architecture to learn phrase translation probabilities. The training is done on phrase translation pairs extracted in the phrase-based translation system. The model re-scores all the phrase-pairs probabilities which are used as additional features in log-linear phrase based translation system. They use WMT'14 translation task to build English/French SMT system coupled with features from encoder-decoder model. With quantitative analysis of the system (on BLEU score), they show that baseline SMT system's performance was improved when RLM was used. Additionally, adding features from proposed Encoder-Decoder architecture increased the performance further suggesting that signals from multiple neural systems indeed add up and are not redundant. They later perform qualitative analysis of their proposed model to investigate the quality of the target phrases generated by model. The target phrases (given a source phrase) proposed by model look more visually appealing than the top target phrases from translation table. They also plot the phrase representations (after dimensionality reduction) on 2-d plane and show that the syntactically and semantically similar phrases are clustered together.

While Cho et al. [6] proposed an end-to-end RNN architecture, they use it only to get additional phrase translation table to be eventually used in the SMT based system. Sutskever et al. [7] gave a more formal introduction to the sequence-to-sequence RNN encoder-decoder architecture. Though their motivation was to investigate the ability of very-deep neural networks at solving seq-to-seq problem, they run their experiments on machine translation task to achieve their goals. They proposed the architecture very similar to Cho et al. [6] with three major architecture changes: 1) they used LSTM cells in encoder and decoder RNN, 2) they trained their system on complete sentence pairs and not just phrases, 3) they used stacked LSTMs (with 4-6 layers) in both decoder and encoder. Finally, they also reverse the source sentences in the training data and train their system on reversed source sentences (keeping the target language sentences in their original order). They don't provide a clear motivation of why they did so, but informally, reversing the source sentence helps in capturing local dependencies around the word from either direction. Their experimentation results (on WMT-14 English/French MT dataset) show that reversing the source sentence achieves higher BLEU score on test set than the model where no reversing was done. Though their model doesn't beat the state-ofthe-art MT system, it achieves performance very close to the latter. Their model doesn't employ any attention methods or bi-directional RNN (which is used by the state-of-the-art system). This suggests that deep models indeed help in seq-to-seq learning with RNN encoder-decoder architecture.

Luong et al. [8] study the effectiveness of NMT systems in spoken language domains by using IWSLT 2015 dataset. They explore two scenarios: NMT adaptation and NMT for low resource translation. For NMT adaptation task, they take an existing state-of-the-art English-German system [9], which consists of 8 individual models trained on WMT data with mostly formal texts (4.5M sentence pairs. They further train on the English-German spoken language data provided by IWSLT 2015 (200K sentence pairs). They show that NMT adaptation is very effective: models trained on a large amount of data in one domain can be finetuned on a small amount of data in another domain. This boosts the performance of an English-German NMT system by 3.8 BLEU points. For NMT low resource translation task, they use the provided English-Vietnamese parallel data (133K sentence pairs). At such a small scale of data, they could not train deep LSTMs with 4 layers as in the English-German case. Instead, they opt for 2-layer LSTM models with 500-dimensional embeddings and LSTM cells. Though their system is little behind the IWSLT baseline (baseline's BLEU score is 27.0 and their model's BLEU score is 26.4), it still shows that NMT systems are quite effective in other domains too, and not just formal texts.

References

1. Neural machine translation. URL: https://en.wikipedia.org/wiki/Neural_machine_translation.

2. A gentle introduction to neural machine translation (in Russian). URL: https://www.machinelearningmastery.ru/introduction-neural-machine-translation/.

3. Neural networks in machine translation: the status quo (in Russian). URL: https://sysblok.ru/nlp/nejronnye-seti-v-mashinnom-perevode-status-kvo/.

4. Ankush Garg, Mayank Agarwal. Machine Translation: A Literature Review. arXiv preprint arXiv:1901.01122. – 17 pp.

5. Nal Kalchbrenner and Phil Blunsom. Recurrent continuous translation models. In Proceedings of the Conference on Empirical Methods in Natural Language Processing, EMNLP 2013. - Seattle, Washington, USA, October 2013. – P. 1700-1709. Association for Computational Linguistics.

6. Kyunghyun Cho, Bart van Merriënboer, Çaglar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1724–1734, Doha, Qatar, October 2014. Association for Computational Linguistics.

7. Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to sequence learning with neural networks. In Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2, NIPS'14, pages 3104-3112, Cambridge, MA, USA, 2014. MIT Press.

8. Minh-Thang Luong and Christopher D. Manning. Stanford neural machine translation systems for spoken language domains. Int. Work. Spok. Lang. Transl. – 2015. – P. 76-79.

9. Minh-Thang Luong, Hieu Pham, and Christopher D Manning. Effective approaches to attention-based neural machine translation. In Empirical Methods in Natural Language Processing (EMNLP). Association for Computational Linguistics. – Lisbon, Portugal, 2015. – P. 1412-1421.