Neural Computing and Applications (2021) 33:15919–15931
https://doi.org/10.1007/s00521-021-06268-0

REVIEW

Neural machine translation: past, present, and future

Shereen A. Mohamed[1] · Ashraf A. Elsayed[1] · Y. F. Hassan[1] · Mohamed A. Abdou[2]

Received: 18 October 2020 / Accepted: 26 June 2021 / Published online: 9 July 2021

Abstract
Deep neural networks (DNN) have achieved great success in several research areas like information retrieval, image processing, and speech recognition. In the field of machine translation, neural machine translation (NMT) has been able to overcome the statistical machine translation (SMT), which has been the dominant technology for a long-term span of time. The recent machine translation approach, which consists of two sub networks named an encoder and a decoder, has gained state-of-the-art performance on different benchmarks and for several language pairs. The increasing interest of researchers in NMT is due to its simplicity compared to SMT which consists of several components tuned separately. This paper describes the evolution of NMT. The different attention mechanism architectures are discussed and the purpose of each. The paper also presents some toolkits that are developed specifically for research and production of NMT systems. The superiority of NMT over SMT is discussed, as well as the problems facing NMT.

Keywords Neural machine translation · Attention mechanism · Self-attentional transformer · Convolutional sequence to sequence

1 Introduction

Neural machine translation has emerged as a new approach in the field of automatic machine translation a few years ago. Despite its short age, NMT has gained great popularity among researchers in this field because of the promising translation results achieved, in addition to the simplicity of its structure [1]. NMT consists of an end-to-end single large neural network. This network contains two sub networks, namely: the encoder and the decoder. To translate a sentence from the source language into its corresponding in the target language, the encoder receives the words in the source sentence and converts them into semantic vector representations. This representation is used by the decoder to generate the target sentence word by word [2].

Since its appearance as a new paradigm in the late 1980s and early 1990s by researchers at IBM T.J. Watson Research Center [3], SMT has been the predominant technique in the field of machine translation. Different approaches have been developed including Phrase-based [4], Syntax-based [5], and Hierarchical phrase-based [6]. These approaches have been dominant for a long time and have been used in many applications because of their superiority over other methods.

NMT has proven to be the first approach able to challenge SMT technologies. At the IWSLT 2015 evaluation expedition, NMT was able to overcome state-of the-art phrase-based machine translation systems on English-German, the language pair famous for its difficulty due to morphology and grammatical differences [1].

NMT has many advantages over SMT [7, 8]:

- The SMT system consists of several components tuned separately. In contrast, The NMT model is a large end-to-end single network that consists of two sub recurrent neural network: the encoder and the decoder.

✉ Shereen A. Mohamed
shereen.nafie@alexu.edu.eg

Ashraf A. Elsayed
ashraf.elsayed@alexu.edu.eg

Y. F. Hassan
y.fouad@alexu.edu.eg

Mohamed A. Abdou
m.abdou@pua.edu.eg

[1] Department of Mathematics and Computer Science, Faculty of Science, Alexandria University, Alexandria, Egypt

[2] Informatics Research Institute, City of Scientific Research and Technological Applications, Alexandria, Egypt

- While the SMT system needs many features that are accurately defined to do the translation, the NMT model depends on a training corpus to learn the translation task, with less or no feature engineering effort by linguists or engineers.
- In contrast to SMT, NMT can seize potential long-distance dependencies and complicated word alignment information
- The NMT model does not require a large memory space, such as those used by the SMT to store a translation model, a reordering model and a language model.

Recently, the machine translation research field is going neural. To give some indication of the speed of change: In 2015, only one neural machine translation system was submitted at the shared task for machine translation organized by the Conference on Machine Translation (WMT). In 2017, almost all submitted machine translation systems were neural.

The paper is organized as follows: Section 2 reviews some of the attempts at developing a NMT system, which have been conducted during the last wave of neural networks research. Also, the integration of neural models in conventional SMT systems. Section 3 describes the emergence of pure NMT systems, and the encoder-decoder structure used in these systems. Section 4 introduces the three architectures of attention mechanism. Section 5 discusses the performance of these architectures on different benchmarks, with a brief description of the benchmarks and the BLEU metric. Section 6 presents some toolkits that are developed specifically for research and production of NMT systems. Section 7 shows some of the challenges facing NMT. Finally, the conclusions are drawn in Sect. 8.

2 Background

The use of neural networks in automatic machine translation attracted the attention of many researchers in the 1990s. In 1997, Forcada and Neco [9] developed a model that could perform simple translation tasks. The model consisted of two feedforward neural networks which they called the encoder and the decoder. The input strings were received symbol by symbol, and an internal representation was produced by the encoder. This representation was passed to the decoder, which in turn produced the corresponding translation. Castaño et al. [10] developed a model to tackle a simple pseudo-natural machine translation task. The model used a simple recurrent network as the basic architecture. The source sentence was received at the input layer word by word, and both preceding and following contexts of the input word were presented to the network. The model generated the translation one word at a time, where the word associated to the neuron with the maximum value was considered the output word. Input and output words were represented by vectors and a specific output neuron was used to mark the end of the translation.

It is noticed that those models were struck in a similar way to the current approaches, but the datasets used in the training were not large enough to support the achieved results. The available computational resources at that time were far less than required.

Despite the decline in the popularity of neural networks, which could not fulfill its promises, especially in the field of industry, the development continued in laboratories and universities.

With the availability of big data, and the recent progress in hardware and software, which includes the development of GUI units and the implementation of libraries that enable the use of GUIs in neural networks processing, interest in the development of neural translation methods has been resurrected. Research has started with the integration of neural-based components into traditional statistical machine translation systems. In 2007, Schwenk [11] used a neural language model for large vocabulary continuous speech recognition. The developed model was evaluated on three different languages, using several large vocabulary speech recognition tasks. The model achieved a significant reduction in word error, and results showed the possibility of using the model in a real-time speech recognizer. In 2012, Schwenk [12] proposed a neural translation model that learns the translation probabilities of pairs in phrase-based SMT, using continuous representations. It worked as follows: First, the input layer receives the one-hot representations of the $n-1$ previous words in a vocabulary. Then, these words are projected on the next layer to form continuous representations of words. One or more hidden layers, which use tanh non-linear activation function, follow this embedding layer. Finally, the output layer estimates the posterior possibilities of all words in the vocabulary, using a softmax normalization. The proposed model could successfully predict the translation for phrase pairs not seen in the training data, also, it could predict a set of the most possible translations given a source phrase. Evaluation of the model, using the English/French IWSLT task, showed an improvement in the BLEU score.

Neural-based approaches sneaked into several SMT components. Other works included replacing other traditional models (*e.g.*, reordering model, pre-ordering model, etc.) with neurals. But, because of the lack of GPU units or the experience to use in the training, this integration of neural models was slowly adopted.

3 Evolution of NMT

The development of pure neural machine translation systems has started with the use of convolutional models and sequence-to-sequence models. In 2013, Kalchbrenner and Blunsom [13] introduced a class of continuous translation models that translates a sentence from a source language into a target language. The class consisted of two models: The first model used convolutional layers to generate a sentence representation based on the continuous representations of the words in the source sentence. This source sentence representation in turn is used to generate words for the target sentence. The second model starts by estimating the length of the target sentence. Then, it generates a representation for the 4-g in the source sentence. From the 4-g representation, the model creates a representation for the sentence that has the predicted length of the target. What is worth mentioning in this work (1) To the best of our knowledge, these models are the first pure NMT models. They were not presented as components to be integrated into SMT to improve performance. Also, no external components from SMT or any other approaches were used in their designs. (2) Although the CNN network was used extensively in image processing, this work provided the first detailed explanation of the CNN's use in texts. The drawback of using CNN in text space, like machine translation, is the inability to pick up dependency between words which are some distance apart, or to learn structural information.

In 2014, Sutskever et al. [14] presented an approach that used a multilayered Long Short-Term Memory (LSTM) to translate sentences from English to French. The first LSTM layers were used to convert the input sequence to a vector of fixed length. This vector was used to generate the output sequence using another LSTM layer. The system consisted of an ensemble of 5 deep LSTMs, and used a simple left-to-right beam-search decoder.

The quantitative analysis the authors carried out showed that the LSTM performed well on long sentences, and learned sentence representations that are sentient to word order. Although they did not provide an explanation for the motive for reversing the word order in the source sentences, the authors showed that doing so further improved the performance of LSTM. Evaluation of the system reported a close to the state-of-the-art performance of the conventional phrase-based machine translation system, and has shown an achievement of 34.8 BLEU on the WMT 14 English to French test set.

In 2014, Cho et al. [15] conducted a study to explore the characteristics of NMT using two models: Recurrent neural network (RNN) Encoder–Decoder, and a proposed gated recursive convolutional neural network. The first model used an RNN with the gated hidden unit in building both the encoder and the decoder. The second model replaced the RNN in the encoder with a proposed gated recursive convolutional neural network. Both models used an encoder-decoder structure, where the encoder converted variable-length source sentences into fixed-length vector representations. Then, the decoder used these representations to generate the variable-length target sentences.

Evaluation of the two models was conducted on French-to-English translation. The results showed high accurate translation of short sentences without unknown words, but this translation performance decreased rapidly with the increase in the length of sentences and the number of unknown words. The authors declared that their most obvious explanatory hypothesis is that using a fixed-length vector to represent an input sentence, whatever its length, sacrifices some important information in the sentence. The results also showed a good performance of the proposed gated recursive convolutional network in learning the grammatical structure of the source sentences.

4 Introducing the attention mechanism

Given a source sentence x, NMT tries to find the target sentence y that maximizes the conditional probability of y given x. training of a NMT system aims to fit a parameterized model to achieve this conditional distribution for sentence pairs in a parallel training corpus. Although convolutional and sequence-to-sequence have proposed to learn this conditional distribution and provided a good translation accuracy, this accuracy was reduced as the length of the input sentence increased. These models have adopted an encoder–decoder approach that compress all the necessary information of a source sentence into a fixed-length vector, which made it difficult for the models to handle long sentences, especially those longer than the training sentences. This problem has been solved with the introduction of the attention mechanism.

Attention mechanism has achieved great popularity and has been used in various fields. In the field of machine translation, the three architectures used are: Stacked RNN with Attention, Self-attentional Transformer, and Fully Convolutional Models (ConvSeq2Seq).

4.1 Stacked RNN with attention

In 2015, and to address the problem of fixed-length vectors, Bahdanau et al. [16] proposed a model that extends the encoder-decoder approach by allowing automatic search for portions of a source sentence, which have relevance to prediction of a target word, without explicitly forming these portions as a hard segment. Instead of encoding a

whole input sentence into a vector of fixed-length, the model converts it into a sequence of vectors. Each time during the decoding process, the decoder searches the input sentence for the words that have the most relevant information to generate the target word. The target word is predicted based on a context vector of all relevant words, and all previously predicted target words. The model is shown in Fig. 1

Assume $x = x_1, x_2, \ldots, x_m, \ldots, x_M$ is a sentence of M words in the source language, and $y = y_1, y_2 \ldots, y_n, \ldots, y_N$ is a sentence of N words in the target language. The introduced model works as follow [17, 18]:

- The input sentence x is passed through a bidirectional RNN encoder to generate a sequence of hidden states. The encoder consists of a forward RNN that reads the sentence from left to right and produce the hidden states $\vec{h}$. And a backward RNN that reads the sentence from right to left and produce the hidden states $\overleftarrow{h}$ where:

$$\overrightarrow{h_m} = \vec{\emptyset}\left(\overrightarrow{h_{m-1}}, x_m\right) \tag{1}$$

$$\overleftarrow{h_m} = \overleftarrow{\emptyset}\left(\overleftarrow{h_{m+1}}, x_m\right) \tag{2}$$

  where $\vec{\emptyset}$ and $\overleftarrow{\emptyset}$ are RNN.
- Both $\overrightarrow{h_m}$ and $\overleftarrow{h_m}$ are concatenated to provide the hidden state h_m for the word x_m

$$h_m = \left[\overrightarrow{h_m}\ ;\ \overleftarrow{h_m}\right] \tag{3}$$

- To predict a word y_n, the model uses the following conditional probability:

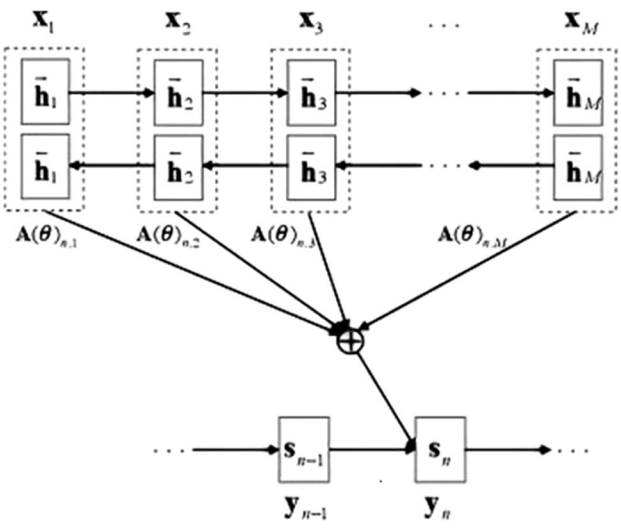

Fig. 1 The architecture of the Stacked RNN with Attention generating the candidate target word yn, given an input sentence (X1, X2, X3, …, XM

$$p(y_n|x, y_{<n}, \theta) = g(y_{n-1}, s_n, c_n, \theta) \tag{4}$$

where $g(.)$ is a non-linear function, s_n is the hidden state corresponding to y_n, c_n is context vector, and θ is a set of model parameters.

- The decoder consists of a single RNN. The hidden states of the decoder are calculated as:

$$s_n = f(s_{n-1}, y_{n-1}, c_n, \theta) \tag{5}$$

  where f is a RNN.
- Now, there are two loose ends: the encoder and the decoder. To tie them together, a context vector c_n is used:

$$c_n = \sum_{m=1}^{M} A(\theta)_{n,m} h_m \tag{6}$$

- $A(\theta)$ is an alignment matrix, in which each element $A(\theta)_{n,m}$ refers to the contribution of the source word x_m in the generation of the target word y_n. It is calculated as follow:

$$A(\theta)_{n,m} = \frac{\exp(a(s_{n-1}, h_m, \theta))}{\sum_{m=1}^{M} \exp\left(a\left(s_{n-1}, h_{m}, \theta\right)\right)} \tag{7}$$

Where $a(s_{n-1},\ h_m,\ \theta)$ measures how well x_m is aligned to y_n.

Although this attention mechanism has been first introduced for machine translation, now it is used widely in neural architectures for several applications in sentiment classification, text summarization, question answering and others. This is due to the following advantages [19]:

- It presents the state-of-the-art for several applications like machine translation, sentiment analysis, and part-of-speech tagging
- It improves both performance and interpretability of the neural models. Neural models have been considered as black-box models. This attention mechanism helps in a better understanding of how these models work to produce the output
- It helps overcome some of the problems that face RNNs, like the decrease in performance that accompanies the increase in the sentence length. The alignment matrix is computed on each word in the input sentence. So, the context vector is not influenced by the sentence length.

The main drawbacks of this architecture are [20, 21]:

- Lack of parallelism: Recurrent models align symbol positions in both the input and the output sequences to steps in computation time. Generation of hidden states at time t requires knowing of the input symbol at time t,

and the hidden state at time t-1. This workflow of recurrent models prevents parallel training of examples.
- Each time the decoder produces an output element, the model must go through the entire input sequence. This consumes considerable time that increases with the lengths of the source and target sequences.
- This architecture is not suitable for "online" tasks where the production of the output elements begins once the input sequence is partially observed.

4.2 Self-attentional transformer

With the aim to preclude sequential computations that exist in recurrent models, Vaswani et al. [22] proposed the self-attentional transformer. Self-attention, also called intra-attention, is the mechanism of attention that generates a representation for a sequence by relating different positions of it. In addition to learning dependencies between inputs and outputs, the self-attentional transformer also learns intra-input and intra-output dependencies. The model is shown in Fig. 2.

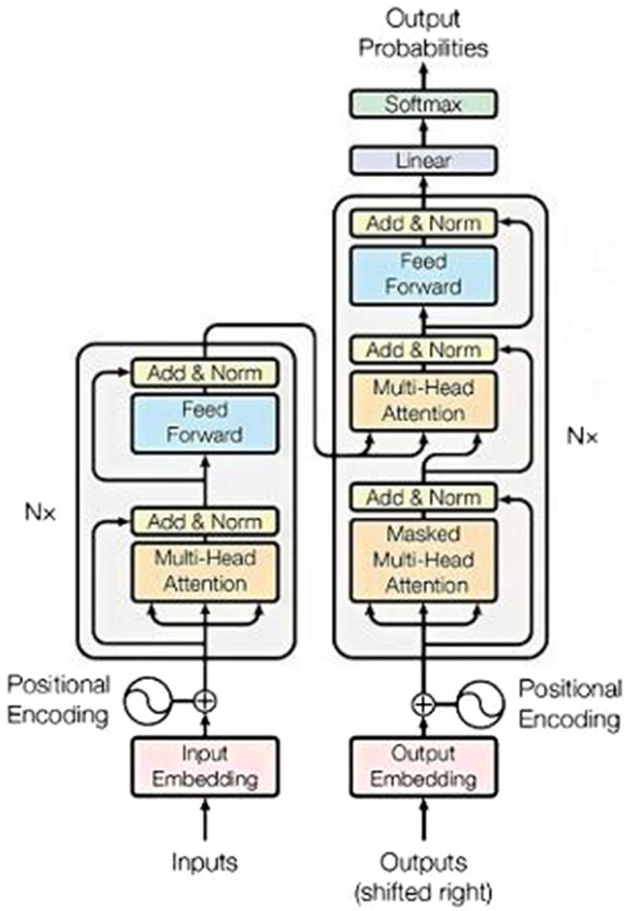

Fig. 2 The architecture of the self-attentional transformer used for learning intra-input and intra-output dependencies

Both the encoder and the decoder consist of a stack of 6 identical layers. Each encoder layer contains two main components: a multi-head self-attention mechanism, and position-wise feed forward network. Each decoder layer contains three main components: a masked multi-head self-attention mechanism, a multi-head self-attention mechanism, and position-wise feed forward network. Residual connections exist around each of the main components, followed by layer normalization.

The model works as follow [23]:

- Both the input and the target sentences are passed through embedding layers to generate word vectors $W_E \in R^{d_m \times V}$, where d_m is the model size, and V is the size of the vocabulary
- These word vectors are then multiplied by a scaling factor of $\sqrt{d_m}$
- To preserve the position information, position encodings of sinusoids are applied and summed to the source and target scaled word vectors

4.3 The encoder layer

- The multi-head self-attention sublayer: implements multi-head dot-product self-attention. Assumes $x = (x_1, \ldots x_T)$ is the input to the layer, and $z = (z_1, \ldots, z_T)$ is the self-attention output, where $x_i, z_i \in R^{d_m}$. In the case of a single-head:
- First, keys, values, and queries are calculated as follow, respectively:

$$k_i = x_i W_K \tag{8}$$

$$v_i = x_i W_V \tag{9}$$

$$q_i = x_i W_Q \tag{10}$$

where W_K, W_V, and W_Q are learnable transformation matrices.

next: Similarity scores between query and key vectors are calculated using the following equation:

$$e_{ij} = \frac{1}{\sqrt{d_m}} q_i \, k_j^T \tag{11}$$

Then, the softmax function is applied on these similarity scores to get the attention coefficients

$$a_{ij} = \frac{\exp e_{ij}}{\sum_{l=1}^{T} \exp e_{il}} \tag{12}$$

An output zi is calculated as the convex combination of attention coefficients with value vectors followed by a linear transformation

$$zi = \left(\sum_{j=1}^{T} a_{ij}\ v_j\right) W_F \tag{13}$$

where W_F is a linear transformation matrix.

- In the case of multi-head, the key, value, and query vectors are split into L vectors. Equations from (1) to (5) are executed in parallel for each of the L vectors. The outputs of Eq. (6) are first concatenated before being linearly transformed by W_F.
- The position-wise feed forward network: This sublayer keeps track of the order of the sentence by generating encodings of the absolute positions of the words in the sentence. The residual connections around the layers retain the positional embeddings across the network.

4.4 The decoder layer

- The masked multi-head self-attention sublayer: performs multi-head self-attention with a modification to prevent positions from appearing at subsequent positions.
- The multi-head self-attention sublayer: performs the inter-attention between the encoder and the decoder. This time, the query vector gets its input from the decoder layer. And both the key and value vectors gets their inputs from the last layer of the encoder.
- The position-wise feed forward network: similar to the encoder

The self-attentional transformer provides two advantages: (1) Parallelization of Sequence-to-sequence: by replacing recurrence with attention and encoding word positions in both input and output sentences, which consequently leads to reduced training time. (2) Reduction of sequential computation: it decreases the number of operations needed to learn dependencies between subsequent words.

Despite the state-of-the-art performance achieved by this architecture, it suffers from the following drawbacks [24, 25]:

- Extraction of unimportant information: it assigns credits to all values regardless of how extent they are correlated to the query. This leads to lack of focus on relevant information and paying attention to irrelevant portions in the context.
- The quadratic increase of computation costs: the self-attention mechanism computes dot products for every pair of words in the source and target sentences. The computational costs increase quadratically with the sentence length, which makes this architecture difficult to use with long sequences.

4.5 Convolutional sequence to sequence

In 2017, Gehring et al. [26], at the Facebook AI Research (FAIR), have proposed this architecture, which is based completely on convolutional neural networks (CNNs). The reasons the researchers choose this network are:

1. In contrast to RNNs, CNNs do not account on the computations of the previous time step, thus, they allow parallel processing of every token in a sentence
2. Despite CNNs encode a representation for a context of fixed-size, this size can be increased using stacks of CNNs layers on top of each other.

Multi-layer CNNs make a hierarchical representation of the input sequence, thus, close input tokens interact at lower layers and far tokens at higher layers. This way, long-range dependencies can be captured by a shorter path. The model is shown in Fig. 3.

Assume $x = (x_1, x_m)$ is a sentence of m words in the source language, and $y = (y_1, y_n)$ is a sentence of n words in the target language. The introduced model works as follow [27, 28]:

- The source sentence x is embedded in a distributional space as $w = (w_1, \ldots, w_m)$ where $w_j \in R^f$ is a column in

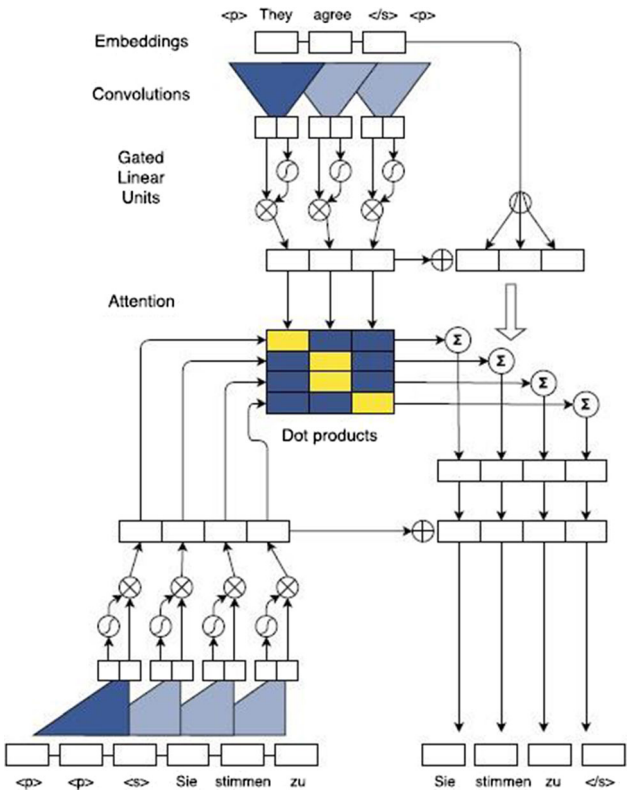

Fig. 3 The architecture of the convolutional Sequence to Sequence generating the target Sentence "Sie stimmen zu", given the input "They agree"

an embedding matrix $D \in R^{V \times f}$ where f is the embedding dimension, and V is the size of the vocabulary

- To keep track of the sentence order, each word position is assigned an embedding as well. A position embedding p = (p_1, …, p_m) that contains absolute positions of words in x is generated, where $p_j \in R^f$.
- Both embeddings are combined to get a word representation e = ($w_1 + p_1$, …, $w_m + p_m$). The same process is applied to the output elements that were already generated by the decoder to produce output element representations q = (q_1, …, q_n) that are being fed back into the decoder
- The architecture employs a shared CNN block structure to compute the intermediate states for both the encoder and the decoder, using a fixed number of input elements (words). Assuming l-th block, the output states of the encoder are $z^l = \left(z_1^l, \ldots, z_m^l\right)$, and the output states of the decoder are $h^l = \left(h_1^l, \ldots, h_n^l\right)$
- Each block has a one dimensional CNN, followed by a non-linearity consisting of gated linear units. Each generated intermediate state holds information over k input elements, where k is the kernel width. To handle the increased number of input elements, blocks can be stacked on top of each other. Non-linearities allow the network to either exploit the whole input field, or to focus on some words.
- Each convolution kernel takes a concatenation of k input elements as input and maps them to one output element. The following layers run on the k elements produced by the previous layer. Simple gating mechanism is carried out on the convolution output by the gated linear units
- Residual connections are added between the input of each convolution and the output of the block. for the l-th layer of the decoder, the residual connections calculate the convolution unit i as:

$$h_i^l = v\left(W^l\left[h_{i-\frac{k}{2}}^{l-1}, \ldots, h_{i+\frac{k}{2}}^{l-1}\right] + b_w^l\right) + h_i^{l-1} \tag{14}$$

Where W^l and b_w^l are the parameters of the convolution kernel.

4.6 The multi-step attention

For each decoder layer, there is a separate attention mechanism. This attention is calculated as follows:

- The current decoder state h_i^l and the embedding of the previous target word g_i are used to compute the decoder state summary

$$d_i^l = w_d^l h_i^l + b_d^l + g_i \tag{15}$$

- the attention weights for a certain state i and a source word j is calculated as the dot-product between d_i^l and each output of the last encoder block u:

$$a_{ij}^l = \frac{\exp\left(d_i^l . z_j^u\right)}{\sum_{t=1}^m \exp\left(d_i^l . z_t^u\right)} \tag{16}$$

- for the current decoder layer, The conditional input is calculated as

$$c_i^l = \sum_{j=1}^m a_{ij}^l \left(z_j^u . e_j\right) \tag{17}$$

Where $\mathbf{e_j}$ is the input word embedding that can give point information about a specific input word. $\mathbf{c_i^l}$ is added to the output of the corresponding decoder layer $\mathbf{h_i^l}$, and used as part of the input to $\mathbf{h_i^{l+1}}$.

The advantages of convolutional Sequence to Sequence are [26]: (1) Parallelization of Sequence-to-sequence: all computations required to be performed during the training can be done in parallel, thus maximizing the utilization of GPUs. (2) Better optimization: the architecture can be better optimized thanks to the presence of a fixed number of gated linear units independent of the input length.

The main drawbacks of this architecture are [22, 29]: (1) the attention functionality depends on the presence of a multi-layer convolution structure, the weighted context from the encoder side would be of less impact in the Absence of this architecture. (2) the Convolutional sequence to sequence performs a set of operations to calculate the dependencies between positions in input and output. The number of operations increases linearly as the distances between these positions increase.

5 Discussion

Attention mechanism has proved superior performances not only in machine translation, but in other fields such as sentiment classification, text summarization and question answering. In the previous section, three attention mechanisms have been represented: Stacked RNN with Attention, Self-attentional Transformer, and Convolutional Sequence to Sequence. Evaluation of these mechanisms has been performed on the English-French and English-German datasets from the WMT 14, and the English-Romanian dataset from the WMT 16. And the BLEU metric has been used to measure translation accuracies.

5.1 Datasets & benchmarks

The workshop on statistical machine translation (WMT) [30] is an annual prestigious scientific gathering for

discussing all emerging trends in the area. The WMT 14 was the ninth workshop; it examined translation between 10 language pairs including the English-French and English-German. Moreover, the WMT 16 examined translation between 12 language pairs including English-Romanian. The WMT 14 English-French dataset [31] provides five parallel corpora: Europarl v7, news commentary, UN corpus, and two crawled corpora (more than 40.5 million sentence pairs). The WMT 14 English-German dataset [31] contains three parallel corpora: Europarl v7, news commentary, and one crawled corpora (about 4.5 million sentence pairs). Finally, WMT 16 English-Romanian dataset [32] contains two parallel corpora: Europarl v8 and a corpus of news articles (about 610,320 sentence pairs).

The BLEU (BiLingual evaluation understudy) [33] metric could be considered as a low-cost and language-independent algorithm for evaluating the quality of an automatic translation. It compares sequences of words of the obtained translation output (candidate translation) with that of the reference translation, to find to what extent they are similar. The more they match, the better the candidate translation is. BLEU's output is a value between 0 and 1. The closer the value to 1, the more similar candidate and reference sentences are.

5.2 Results comparison

With the motivation to tackle the problem of using a fixed-size vector to encode the source sentence, Bahdanau et al. [16] proposed the Stacked RNN with Attention. To analyze the model performance, they trained and tested it using sentences of length up to 30 words. Then, sentences of length up to 50 words. And finally, sentences of length greater than 50 words. Evaluation of the proposed model on the English-French parallel corpora from the WMT 2014 translation task showed a significant improvement in comparison with the basic encoder-decoder. This improvement was obvious, especially with long sentences. Also, the results showed a translation performance comparable to existing phrase-based statistical machine translation systems.

The Convolutional Sequence to Sequence model proposed by Gehring et al. [26] showed a remarkable superiority over the model of bahdanau et al. Also, the model achieved a new state-of-the-art when evaluated on the WMT 16 English-Romanian dataset, and a comparable BLEU score on the WMT 14 English-German dataset. Compared to the RNN stack with attention, the Convolutional sequence to sequence provides a shorter path to capture dependencies between words which are some distance apart. Also, the computational complexity is far less.

Vaswani et al. [22] were able to avoid the drawbacks of RNN and CNN by developing a model based solely on attention mechanisms. The model achieved a comparable results on the WMT 14 English-German dataset, and a state-of-the-art BLEU score on the WMT 14 English-French translation task. It requires a training time less than the models based on CNN or RNN layers. In comparison to the Convolutional sequence to sequence, the Convolutional sequence to sequence consumes time to calculate the dependencies between positions in input and output. This time increases linearly as the distances between these positions increase. While, the calculation of these dependencies depends on a fixed number of operations in Self-attentional Transformer. Tables 1 and 2 show the performance of the attention mechanisms presented in [16, 22, and 26] through the WMT14 English-French dataset and WMT14 English-German dataset, respectively. From these tables, it could be concluded that Vaswani [22] surpassed the other models in both execution time and RNN performance. On the other hand, Gehring [26] was not too far from Vaswani [22] in BLEU score in the English-French dataset. Furthermore, Gehring [26] analyzed an English-Romanian dataset and achieved 30.02 BLEU score.

6 Toolkits

There is a wide spread of toolkits that has been mainly implemented to support research, development, and deployment of neural machine translation systems. This number of toolkits is increasing as more are being developed. According to the tally by nmt-list [34], there are 55 toolkits. Some of the hopeful toolkits are OpenNMT, XNMT, Nematus, SOCKEYE.

6.1 OpenNMT

OpenNMT [35], the open-source toolkit launched in December 2016, has been developed for neural machine translation and neural sequence modeling. There are three implementations of OpenNMT for both academic and industrial purposes. They have been developed taking into consideration simplicity of use, ease of extension, and efficiency and state-of-the-art accuracy. These implementations are:

- OpenNMT-lua: the original project developed with LuaTorch.
- OpenNMT-py: a clone of OpenNMT-lua that has been developed using PyTorch. This implementation has been initially created by the Facebook AI research team, and is especially suited for research.
- OpenNMT-tf: another alternative of OpenNMT-lua written with TensorFlow. It benefits from the

Table 1 Performance of the three attention mechanisms (models) through the WMT 14 English-French benchmark in BLEU

Mechanisms (Models)	BLEU scores on WMT 14 English-French	Comments
Bahdanau [16] Up to 30 words	21.50	It is robust to the sentence length It does not require encoding the whole input sentence, but only the parts related to a particular word
Bahdanau [16] Up to 50 words	26.75	It consumes more time, to train the model on sentences more than 50 words, until the performance on the validation dataset stops improving
Bahdanau [16] More than 50 words	28.45	It provides a shorter path, Compared to the Bahdanau [16], to capture dependencies between words which are some distance apart
Gehring [26]	40.51	The computational complexity is far less
Vaswani [22]	41	It is able to avoid the drawbacks of RNN used in Bahdanau [16] It requires less training time

Table 2 Performance of Gehring [26] and Vaswani [22] attention mechanisms (models) through the WMT 14 English-German benchmark in BLEU

Mechanisms (Models)	BLEU scores on WMT 14 English-German	Comments
Gehring [26]	25.16	Time needed to calculate the dependencies between positions in input and output increases linearly as the distances between these positions increase, which makes it a hard work to learn dependencies between positions at distance
Vaswani [22]	28.4	The calculation of dependencies between positions in input and output depends on a fixed number of operations

tensorFlow features to serve high-performance model and focus on large-scale experiments.

6.2 XNMT [36]: The extensible neural machine translation toolkit

XNMT is a toolkit that focuses on modular code design, and enables rapid iteration in research, and replicable and dependable results. It can be used in the fields of: Machine translation, speech recognition, and multi-tasked machine translation/parsing. What distinguishes this toolkit from others is the reduced time to convert the idea into a practical experimental environment, testing of these ideas using a large number of parameters, and production of correct and reliable search results. The toolkit has been developed taking into consideration:

- The use of modular code design. So, it is easy to modify code and swap between the different parts of the model.
- Implementation of the toolkit in Python, the standard programming language in the research community.
- The use of DyNet framework, which makes it possible to implement complex networks with dynamic structure, handle batch operations, or rely on autobatching.
- Support of standard NMT models, optimization techniques, multi-task learning, and encoders for speech

6.3 Nematus [37]

An open-source toolkit that has been developed on the basis of the dl4mt-tutorial [38]. It has been implemented in Python, setting a high priority in translation accuracy, usability, and extensibility.

Although Nematus has implemented its encoder-decoder architecture with an attention mechanism similar to that of Bahdanau et al., it has introduced some modifications, like:

- Initialization of the decoder hidden state with the mean of the source annotation
- Implementation of a new conditional GRU with attention.

- Use of tanh non-linearity function instead of maxout in the hidden layer before the softmax layer in the decoder.
- Remove of the additional biases from word embedding layers in both the encoder and the decoder
- Simpler implementation of the decoder Look, Generate, Update process
- Instead of using a single word embedding at each time step in the input sentence, a multiple features representation is used. The final embedding of the input sentence is the concatenation of the embeddings of each feature.

Nematus has been used in the development of high-performance systems to some translation tasks at WMT and IWSLT, as well as in the training of WIPO.

6.4 Sockeye [39]

A free and open-source sequence-to-sequence toolkit that is Written in Python and built on MXNET library. SOCKEYE has been developed to be an appropriate platform for the experiments carried out by researchers, as well as a production software-ready tool for models training and applying. As stated by the authors, SOCKEYE is the only toolkit that comprises implementations for the three attention mechanism techniques: the attention mechanism by [Schwenk, 2012, Kalchbrenner and Blunsom 2013, Sutskever et al. 2014, Bahdanau et al. 2014, Luong et al. 2015], attention transformer by [Vaswani et al. 2017], and fully convolutional networks by [Gehring et al. 2017].

SOCKEYE offers several optimizers, normalization and regularization techniques. In a comparison of SOCKEYE against other NMT toolkits, it has achieved competitive BLEU scores using English–German and Latvian–English datasets from the 2017 Conference on Machine Translation (WMT).

SOCKEYE offers a range of features, to name a few:

- Weight tying: sockeye enables sharing of weights for efficient language modeling and reduction of memory consumption
- RNN attention types: in the attention mechanism by Bahdanau et al. a score function is calculated to generate the attention vector. SOCKEYE gives a wide range of functions to compute that score function.
- RNN coverage models: in contrast to other approaches that keep track of the source-language coverage, NMT does not track the portions of the input sentence that have been translated. This results in over-generation and under-generation problems. Recently, some coverage models have been proposed. SOCKEYE includes implementations of different variants of the proposed coverage modeling
- Optimizers: SOCKEYE makes use of different optimizers from MXNET's library, like stochastic gradient descent (SGD) and Adam. besides, SOCKEYE has its own implementation of other optimizers, like the Eve optimizer, which is an extension of the Adam optimizer.

6.5 Marian [40]

An independent and efficient neural machine translation toolkit written entirely in pure c++, based on dynamic computation graphs. It has been developed at the University of Edinburgh, and at the Adam Mickiewicz University in Poznan, with the goal of providing a research tool capable of defining state-of-the-art systems as well as producing models that can be deployed across different devices. Marian has been able to occupy a distinguished position among other open-source NMT toolkits by providing many advantages:

- Extensible Encoder-Decoder Framework: It provides the possibility to incorporate various encoders and decoders (for example a RNN-based encoder with a Transformer decoder)
- Efficient Meta-algorithms: it provides implementations of several effective meta-algorithms like multi-device (GPU or CPU) training and ensembling of diverse models
- Automatic batch size adaptation: it adjusts the batch size with focus on available memory to get both the maximum speed and memory usage. This ensures that the memory budget chosen during training is not exceeded.

Marian has been used as the main translation and training engine of WIPO, as well as in the development of several European projects.

6.6 Tensor2tensot [41]

An open source library of deep learning models that has been developed by Google. It is appropriate for neural machine translation and has been designed with an emphasis on making deep learning research faster and attainable. Tensor2Tensor offers a host of features, to mention some:

- Researchers can use different devices (CPU, GPU, and TPU), single or multiple, locally or in the cloud, to train their models with no or negligible amount of device-specific code
- As the development of Tensor2Tensor began with an emphasis on neural machine translation, the library

contains many of the most effective NMT models and standard datasets
- It provides support (models and datasets) for tasks in research areas other than neural machine translation as well.
- The library provides a great deal of consistency across models and problems. a single model can be tried on several problems, and several models can be tried on a single problem

6.7 Fairseq [42]

An open-source sequence modeling library based on PyTorch that has been developed to help researchers train models for text generation tasks such as machine translation, language modeling, and text summarization. FAIRSEQ has been developed taking into consideration speed, extensibility, and Benefits for both research and production. It provides a set of features:

- Extensibility: it can be extended with user-supplied plug-ins. This offers the possibility to try new ideas while making use of existing components
- Reproducibility and forward compatibility: results can be reproduced in case the training is interrupted and resumed as the checkpoints keep all the needed data about the model, optimizer, and dataloader. also, Models that have been trained on the old versions of the toolkit will continue to run on new versions
- Batching: source and target sequences are grouped in mini-batches according to the length of the sequence. Thus, padding can be minimized
- Multi-GPU training: a model can be trained using multiple GPUs by making a copy of the model for each GPU to process sub-batch of data. The toolkit offers solutions to the idle time problem where most GPUs wait for the slower ones to finish their work.
- Mixed precision: FAIRSEQ supports both full precision floating point (FP32) and half precision floating point (FP16) at training and inference. The toolkit performs dynamic loss scaling to avoid the problem of underflows of activations and gradients resulting from limited precision of FP16

6.8 Neural monkey

Neural Monkey [43] is an open-source toolkit that has been developed for neural machine translation and neural sequence modeling. It has been developed using the Tensorflow machine learning library with the aim of collecting implementations of modern deep learning methods in diverse fields, with a main focus on NMT. The toolkit has been developed taking into consideration:

- The use of abstract building blocks. so, the user does not need to delve into the details of the implementations
- Support of the quick building of complex models with multiple encoders and decoders.
- Deploying of trained models either to process batch data or as a web service.
- Run of models both on CPU and GPU with a minimal requirements that can be easily installed
- Keeping of the overall structure of a model in an easy-to-read file

6.9 Challenges facing neural machine translation

Although its promising results and rapid adoption in deployments by Google, Systran, and WIPO, neural machine translation faces several challenges [44]:

- System ambiguity: NMT systems are extremely less explainable. Research is needed to discover the reasons why training data make these systems choose a particular word during decoding.
- Domain mismatch: The same word can have different translations according to different domains. Therefore, domain adaptation is a necessary issue to be considered when developing machine translation. The performance of a NMT system becomes dramatically worse when translating out-of-domain sentences. The current approach to deal with this issue is to train the system using a general domain training data, then, train it using in-domain training data. But, more research is needed to find out other approaches.
- Amount of training data: NMT gives highly accurate results as long as the training corpus is large. Whenever the training data size is reduced, the accuracy decreases sharply.
- Rare words: Because of its limited size, a NMT vocabulary contains only frequently used words. NMT systems exhibit weakness in translating out-of-vocabulary or low-frequency words. Several approaches have been developed to address this challenge, but it is still an open point of research
- Long sentences: NMT systems give good results when translating short sentences, they maintain a comparable level of accuracy up to a sentence length of about 60. As the sentence becomes long, the translation quality decreases

7 Conclusion

The use of DNN in machine translation has begun a few years ago; nevertheless, it has achieved great success and high accuracy, surpassing many other approaches including SMT, which has been the dominant approach of machine translation for a long period of time. To translate a sentence from a source language to a target language, the NMT approach employs 2 sub networks, namely: the encoder and the decoder. The encoder converts the source sentence words into some representation, which is used by the decoder in turn to produce the target sentence. The network models used in the construction of the Encoder/Decoder have evolved from the use of the convolutional networks to the use of the bidirectional RNN, which has been able to overcome the problem of fixed-length vectors, where the sentence was represented, regardless of its length, by a vector of fixed length.

The attention mechanism has been able to improve the accuracy of translation, where in each time step the decoder produces a target word depending on articular words in the source sentence. Research in the NMT field has shown some problems that face this technique, like: large vocabulary, rare words, long distance, and domain mismatch. Despite the proposed solutions to these problems, research is still continuing to find better solutions.

References

1. Bentivogli L, Bisazza A., Cettolo M, Federico M. (2016). Neural versus phrase-based machine translation quality: a case study. arXiv preprint AxXiv:1608.04631
2. Zhang B, Xiong D, Su J, Duan H (2017) A context-aware recurrent encoder for neural machine translation. IEEE/ACM Transact on Aud, Speech, Lang Process 25(12):2424–2432
3. Almansor, E. H. (2018). Translating Arabic as low resource language using distribution representation and neural machine translation models (Doctoral dissertation).
4. Moussallem D, Wauer M, Ngomo ACN (2018) Machine translation using semantic web technologies: A survey. J Web Semant 51:1–19
5. Williams P, Sennrich R, Post M, Koehn P (2016) Syntax-based statistical machine translation. Synth Lectur on Human Lang Technolog 9(4):1–208
6. Satpathy S, Mishra, S. P, Nayak, A. K. (2019, May). Analysis of Learning Approaches for Machine Translation Systems. In 2019 International Conference on Applied Machine Learning (ICAML) (pp. 160–164). IEEE
7. Wang X, Tu Z, Zhang M (2018) Incorporating Statistical Machine Translation Word Knowledge Into Neural Machine Translation. IEEE/ACM Transact on Audio, Speech, Lang Process 26(12):2255–2266
8. Yang Z, Chen W, Wang F, Xu B (2018) Generative adversarial training for neural machine translation. Neurocomputing 321:146–155
9. Forcada M L, Ñeco R P. (1997, June). Recursive hetero-associative memories for translation. In International Work-Conference on Artificial Neural Networks. 453–462. Springer, Berlin
10. Castano, M. A., Casacuberta, F., Vidal, E. (1997). Machine translation using neural networks and finite-state models. Theoretical and Methodological Issues in Machine Translation (TMI), 160–167
11. Schwenk H (2007) Continuous space language models. Comput Speech Lang 21(3):492–518
12. Schwenk H. (2012). Continuous space translation models for phrase-based statistical machine translation. Proceedings of COLING 2012: Posters, 1071–1080.
13. Kalchbrenner N, Blunsom P. (2013). Recurrent continuous translation models. In Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing. 1700–1709
14. Sutskever I, Vinyals, O, Le, Q. V. (2014). Sequence to sequence learning with neural networks. In Advances in neural information processing systems. 3104–3112
15. Cho K, Van Merriënboer B, Bahdanau D, Bengio Y. (2014). On the properties of neural machine translation: Encoder-decoder approaches. arXiv preprint arXiv:1409.1259
16. Bahdanau D, Cho K, Bengio Y. (2014). Neural machine translation by jointly learning to align and translate. arXiv preprint arXiv:1409.0473
17. Cheng Y. (2019). Agreement-based joint training for bidirectional attention-based neural machine translation. In Joint Training for Neural Machine Translation. 11–23. Springer, Singapore
18. Choi H, Cho K, Bengio Y (2018) Fine-grained attention mechanism for neural machine translation. Neurocomputing 284:171–176
19. Hu D (2019) An introductory survey on attention mechanisms in NLP problems. In Proceedings of SAI Intelligent Systems Conference. 432–448. Springer, Cham.
20. Dhanani F, Rafi, M (2020) Attention Transformer Model for Translation of Similar Languages. In Proceedings of the Fifth Conference on Machine Translation. 387–392
21. Raffel C, Luong M. T, Liu P. J, Weiss R. J, Eck D (2017) Online and linear-time attention by enforcing monotonic alignments. In International Conference on Machine Learning. 2837–2846. PMLR
22. Vaswani A., Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez A. N, Polosukhin I (2017) Attention is all you need. In Advances in Neural Information Processing Systems. 5998–6008
23. Sachan D. S, Neubig G (2018). Parameter sharing methods for multilingual self-attentional translation models. arXiv preprint arXiv:1809.00252
24. Zhao G, Lin J, Zhang Z, Ren X, Su Q, Sun X (2019) Explicit sparse transformer: Concentrated attention through explicit selection. arXiv preprint arXiv:1912.11637
25. Tsunoo E, Kashiwagi Y, Watanabe S (2020) Streaming Transformer ASR with Blockwise Synchronous Beam Search. arXiv preprint arXiv:2006.14941
26. Gehring J, Auli M, Grangier D, Yarats D, Dauphin Y. N (2017) Convolutional sequence to sequence learning. In Proceedings of the 34th International Conference on Machine Learning-Volume 70. 1243–1252. JMLR. org
27. Wu Y. C, Yin F, Zhang X. Y, Liu L, Liu C. L (2018) Scan: Sliding convolutional attention network for scene text recognition. arXiv preprint arXiv:1806.00578
28. Wang L, Yao J, Tao Y, Zhong L, Liu W, Du Q (2018) A reinforced topic-aware convolutional sequence-to-sequence model for abstractive text summarization. arXiv preprint arXiv:1805.03616

29. Yin W, Schütze H (2018) Attentive convolution: Equipping cnns with rnn-style attention mechanisms. Trans Assoc Computat Ling 6:687–702
30. http://statmt.org.http://statmt.orgLast access 4 February 2021
31. http://www.stamt.org/wrnt14/translation-task.html Last access 4 February 2021
32. http://www.stamt.org/wrnt14/translation-task.htmlLast access 4 February 2021
33. Zakraoui J, Saleh M, Al-Maadeed S, AlJa'am JM (2020) Evaluation of Arabic to English Machine Translation Systems. In 2020 11th International Conference on Information and Communication Systems (ICICS). 185–190. IEEE
34. http://github.com/jonsafari/nmt-list Last access 4 February 2021
35. http://opennmt.netLast access 4 February 2021
36. Neubig G, Sperber M, Wang X, Felix M, Matthews A, Padmanabhan S, Hewitt J (2018) XNMT: The extensible neural machine translation toolkit. arXiv preprint arXiv:1803.00188
37. Sennrich R, Firat O, Cho K, Birch A, Haddow B, Hitschler J, Nădejde M (2017) Nematus: a toolkit for neural machine translation. arXiv preprint arXiv:1730.04357
38. http://github.com/nyu-di/di4mt-tutorial Last access 4 February 2021
39. Hieber F, Domhan T, Denkowski M, Vilar D, Sokolov A., Clifton A., Post M (2017) Sockeye: A toolkit for neural machine translation. arXiv preprint arXiv:1712.05690
40. Junczys-Dowmunt M, Grundkiewicz R, Dwojak T, Hoang H, Heafield K, Neckermann T, Birch A (2018) Marian: Fast neural machine translation in C++. arXiv preprint arXiv:1804.00344
41. Vaswani A, Bengio S, Brevdo E, Chollet F, Gomez A. N, Gouws S, Uszkoreit J (2018) Tensor2tensor for neural machine translation. arXiv preprint arXiv:1803.07416
42. Ott M, Edunov S, Baevski A, Fan A, Gross S, Ng N, Auli M (2019) fairseq: A fast, extensible toolkit for sequence modeling. arXiv preprint arXiv:1904.01038
43. Helcl J, Libovický J (2017) Neural monkey: An open-source tool for sequence learning. The Prague Bullet Mathemat Ling 107(1):5
44. Koehn P, Knowles R. (2017). Six challenges for neural machine translation. arXiv preprint arXiv:1760.03872