Neural Machine Translation

Mustak Ahamed Yadiki
University of New Haven
(myadi1@unh.newhaven.edu)

Poojitha **Mandapati**
University of New Haven
(pmand12@unh.newhaven.edu)

Rahulsai Somepalli
University of New Haven
(rosme1@unh.newhaven.edu)

Abstract

Machine translation (MT) is an important sub-field of natural language processing that aims to translate natural languages using computers. In recent years, end-to-end neural machine translation (NMT) has achieved great success and has become the new mainstream method in practical MT systems. Here, we first provide a broad review of the methods for NMT and focus on methods relating to architectures, decoding, and data augmentation. Then we summarize the resources and tools that are useful for researchers. Finally, we conclude with a discussion of possible future research directions.

Neural Machine Translation (NMT) using an encoder-decoder architecture is a cutting-edge approach in the field of natural language processing that aims to automate the translation of text from one language to another. This methodology leverages artificial neural networks to achieve more accurate and contextually rich translations compared to traditional rule-based or statistical machine translation methods. One of the key advantages of NMT with encoder-decoder architecture is its ability to handle variable-length input and output sequences, allowing for more flexibility in translating diverse texts. Additionally, the model can learn complex patterns and relationships within the data, enabling it to produce more fluent and contextually accurate translations.

GitHub Link: https://github.com/RahulSomepalli/Neural-machine-translation

Introduction:

Neural Machine Translation (NMT) is a state-of-the-art approach to language translation that utilizes artificial neural networks to automatically and contextually translate text from one language to another. This technology has significantly improved the quality and fluency of machine translation compared to traditional rule-based or statistical methods.

The core of NMT systems is based on deep learning models, particularly Recurrent Neural Networks (RNNs) or Transformer architectures. These models are trained on large parallel corpora, which consist of pairs of sentences in the source and target languages. During training, the neural network learns to map input sequences to output sequences, capturing complex patterns and dependencies in language.

Key characteristics of Neural Machine Translation (NMT):

End-to-End Learning: NMT systems learn to translate directly from source to target language without relying on intermediate representations or linguistic rules. This end-toend approach allows for more natural and contextually accurate translations.

Context Awareness: NMT models consider the entire context of a sentence, allowing them to capture long-range dependencies and understand the nuances of language. This context awareness contributes to improved translation quality.

Encoder-Decoder Architecture: NMT models typically consist of an encoder network that processes the input sentence and a decoder network that generates the corresponding translation. This architecture enables the model to effectively capture the meaning of the source text and produce coherent translations.

Attention Mechanism: Many NMT models incorporate attention mechanisms, which enable the model to focus on different parts of the source sentence while generating the translation. This attention mechanism improves the handling of long sentences and helps produce more accurate translations.

Continuous Improvement: NMT systems can be fine-tuned and improved over time as more high-quality parallel data becomes available. Continuous training allows the model to adapt to new language patterns, slang, and evolving linguistic trend. It uses neural network models to learn a statistical model for machine translation. The key benefit to the approach is that a single system can be trained directly on source and target text, no longer requiring the pipeline of specialized systems used in statistical machine learning.

Unlike the traditional phrase-based translation system which consists of many small sub-components that are tuned separately, neural machine translation attempts to build and train a single, large neural network that reads a sentence and outputs a correct translation.

Proposed Methodology

The proposed methodology for Neural Machine Translation (NMT) typically involves several key steps, from data preparation to model training and evaluation. Here's a generalized outline of the methodology:

Data Collection and Preprocessing:

Collect parallel corpora: Gather a sizable dataset containing pairs of sentences in the source and target languages. This data is crucial for training and evaluating the NMT model.

Preprocess the data: Clean and tokenize the text, handle special characters, and perform any necessary language-specific preprocessing. Ensure that the data is aligned correctly.

Data Splitting:

Divide the dataset into training, validation, and test sets. The training set is used to train the model, the validation set helps tune hyperparameters, and the test set evaluates the final performance of the model.

Word Embeddings:

Convert words into continuous vector representations using pre-trained word embeddings like Word2Vec, GloVe, or train embeddings specifically for your dataset. These embeddings capture semantic relationships between words.

Model Architecture:

Choose an appropriate NMT architecture. Popular choices include Recurrent Neural Networks (RNNs), Long Short-Term Memory networks (LSTMs), or Transformer models. The Transformer architecture, introduced by Vaswani et al., has gained widespread popularity for its efficiency and scalability.

Encoder-Decoder Design:

Design the encoder-decoder architecture. The encoder processes the source language input, while the decoder generates the target language output. Include mechanisms like attention to allow the model to focus on relevant parts of the source sentence during translation.

Loss Function:

Define a suitable loss function, typically a sequence-to-sequence loss like crossentropy. This measures the difference between the predicted translations and the actual target translations.

Training:

Train the model using the training dataset. Adjust the model's parameters (weights and biases) iteratively to minimize the defined loss function. Monitor the model's performance on the validation set to avoid overfitting.

Hyperparameter Tuning:

Experiment with different hyperparameters, such as learning rate, batch size, and model architecture, to optimize the model's performance. Use the validation set to fine-tune these parameters.

Evaluation:

Assess the model's performance on the test set using appropriate evaluation metrics such as BLEU (Bilingual Evaluation Understudy) or TER (Translation Edit Rate). These metrics measure the similarity between the model's translations and reference translations.

Post-Processing:

Implement any necessary post-processing steps to improve the fluency and correctness of the generated translations. This may involve handling punctuation, capitalization, or language-specific idiosyncrasies.

Deployment:

Once satisfied with the model's performance, deploy it for practical use in a production environment. This could involve integrating the model into an application, website, or other translation services.

Encoder-Decoder Model:

Multilayer Perceptron neural network models can be used for machine translation, although the models are limited by a fixed-length input sequence where the output must be the same length.

These early models have been greatly improved upon recently through the use of recurrent neural networks organized into an encoder-decoder architecture that allow for variable length input and output sequences.

An encoder neural network reads and encodes a source sentence into a fixed-length vector. A decoder then outputs a translation from the encoded vector. The whole encoder–decoder system, which consists of the encoder and the decoder for a language pair, is jointly trained to maximize the probability of a correct translation given a source sentence.

Key to the encoder-decoder architecture is the ability of the model to encode the source text into an internal fixed-length representation called the context vector. Interestingly, once encoded, different decoding systems could be used, in principle, to translate the context into different languages.

Almost all neural machine translation models employ the encoder-decoder framework. The encoder-decoder framework consists of four basic components: the embedding layers, the encoder and decoder networks,

and the classification layer. shows a typical autoregressive NMT model using the encoder-decoder framework, which we shall use as an example. ‘‘<bos>’’ and ‘‘<eos>’’ are special symbols that mark the beginning and ending of a sentence, respectively.

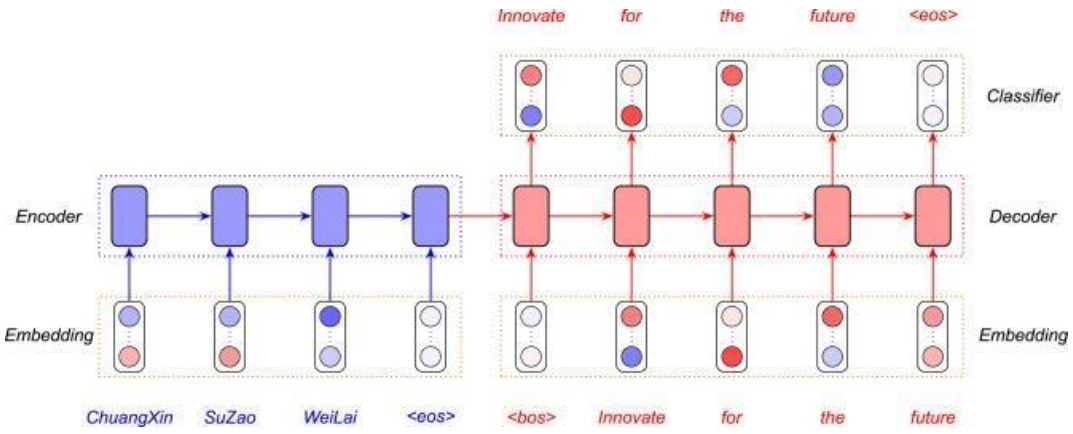
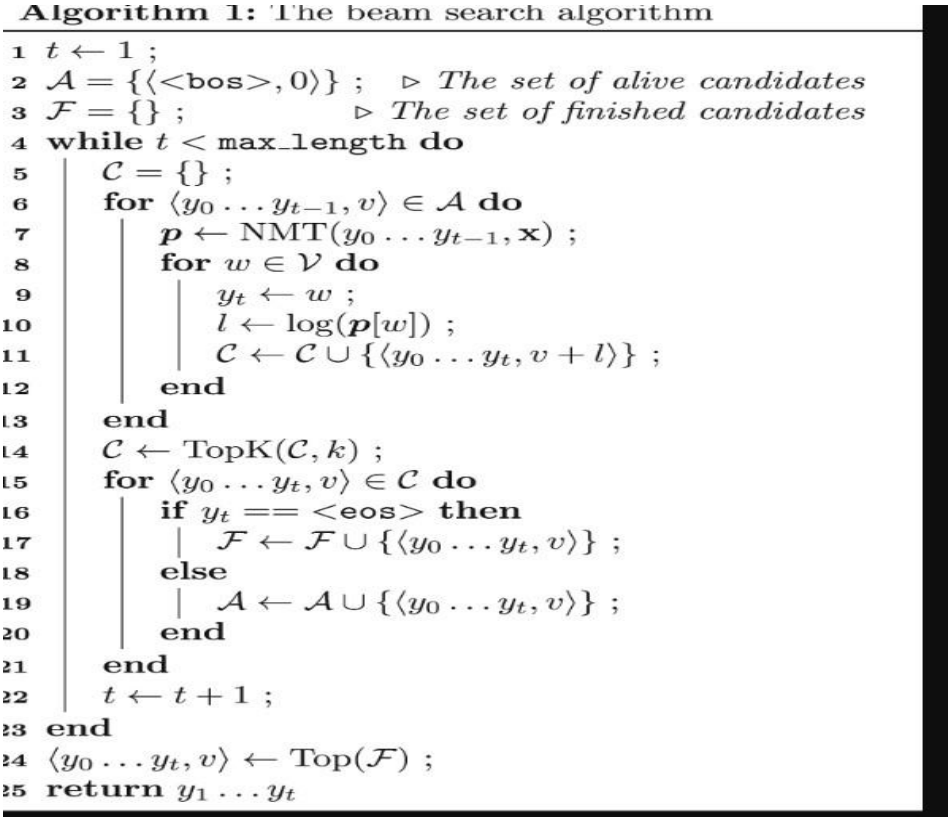


Figure 1:

Example:

Given an NMT model and a source sentence x, how to generate a translation from the model is an important problem. Ideally, we would like to find the target sentence y which maximizes the model prediction p(y|x = x;0) as the translation. However, due to the intractably large search space, it is impractical to find the translation with the highest probability. Therefore, NMT typically uses local search algorithms such as greedy search or beam search to find a local best translation. Beam search Alogrithm is a classic local search algorithm which have been widely used in NMT. Previously, beam search have been successfully applied in SMT. The beam search algorithm keeps track of k states during the inference stage. Each state is a tuple (y0....yt,v) where y0...yt is a candidate translation, and v is the logprobability of the candidate. At each step, all the successors of all k states are generated, but only the top-k successors selected. The algorithm usually terminates when the step exceed a pre-defined value or k full translation are found. It should be noted that the beam search will degrade into the greedy search if k=1



Algorithm 1: The beam search algorithm

$t \leftarrow 1$;
$\mathcal{A} = \{\langle \texttt{<bos>}, 0 \rangle\}$; ▷ *The set of alive candidates*
$\mathcal{F} = \{\}$; ▷ *The set of finished candidates*
while $t <$ `max_length` **do**
  $\mathcal{C} = \{\}$;
  for $\langle y_0 \dots y_{t-1}, v \rangle \in \mathcal{A}$ **do**
    $\boldsymbol{p} \leftarrow \mathrm{NMT}(y_0 \dots y_{t-1}, \mathbf{x})$;
    for $w \in \mathcal{V}$ **do**
      $y_t \leftarrow w$;
      $l \leftarrow \log(\boldsymbol{p}[w])$;
      $\mathcal{C} \leftarrow \mathcal{C} \cup \{\langle y_0 \dots y_t, v + l \rangle\}$;
    end
  end
  $\mathcal{C} \leftarrow \mathrm{TopK}(\mathcal{C}, k)$;
  for $\langle y_0 \dots y_t, v \rangle \in \mathcal{C}$ **do**
    if $y_t ==$ <eos> **then**
      $\mathcal{F} \leftarrow \mathcal{F} \cup \{\langle y_0 \dots y_t, v \rangle\}$;
    else
      $\mathcal{A} \leftarrow \mathcal{A} \cup \{\langle y_0 \dots y_t, v \rangle\}$;
    end
  end
  $t \leftarrow t + 1$;
end
$\langle y_0 \dots y_t, v \rangle \leftarrow \mathrm{Top}(\mathcal{F})$;
return $y_1 \dots y_t$

Figure 2:

Technical Details

Neural Machine Translation (NMT) using an encoder-decoder model is a sequenceto-sequence architecture designed to translate a sequence of words from one language to another. The model consists of two main components: an encoder that processes the input sequence, and a decoder that generates the output sequence.

1. Encoder:

The encoder takes the input sequence (source language) and transforms it into a fixed-dimensional context vector. The encoder's architecture can vary, but commonly used components include:

Embedding Layer: Converts words into continuous vector representations.

Recurrent Layers (e.g., LSTM or GRU): Captures sequential dependencies in the input sequence. Each hidden state represents information about the sequence up to that point.

Bidirectional Encoding: Some models use bidirectional LSTMs/GRUs to process the input sequence both forward and backward. This allows the encoder to capture information from both past and future context.

Encoder Hidden States: The final hidden states or a weighted combination of all hidden states represent the context vector, summarizing the input sequence.

2. Decoder:

The decoder takes the context vector from the encoder and generates the output sequence (target language). Its architecture includes:

Initial State: The context vector from the encoder serves as the initial hidden state for the decoder.

Embedding Layer: Converts the target language words into continuous vector representations.

Recurrent Layers (e.g., LSTM or GRU): Captures sequential dependencies in the output sequence. It generates hidden states that represent the evolving context for generating each word.

Attention Mechanism: Enables the model to focus on different parts of the source sequence during the decoding process. Attention helps the model align words in the source and target languages, improving the translation quality.

Output Layer: Produces a probability distribution over the target vocabulary for each time step. Sampling from this distribution generates the translated words.

3. Training:

During training, the model is optimized to minimize the difference between its predicted translations and the actual target translations. This involves:

Loss Function: Typically, a sequence-to-sequence loss function like cross-entropy is used to measure the dissimilarity between predicted and target sequences.

Backpropagation and Optimization: Gradients are computed with respect to the loss, and optimization algorithms (e.g., Adam, SGD) are employed to update the model parameters.

4. Inference:

During inference, the trained model is used to generate translations for new input sequences:

Greedy Decoding: At each decoding step, the model selects the word with the highest probability as the next word in the sequence.

Beam Search: A more sophisticated decoding strategy that considers multiple possible sequences and selects the one with the highest overall probability.

5. Post-Processing:

Post-processing steps may be applied to improve the quality of the generated translations, such as handling punctuation, capitalization, or language-specific idiosyncrasies.

These technical details provide a high-level overview of the architecture and training process for an encoder-decoder model in Neural Machine Translation. Implementation specifics can vary based on the chosen framework (e.g., TensorFlow, PyTorch) and model configuration.

Training and evaluating a Neural Machine Translation (NMT):

NMT typically uses maximum log-likelihood (MLE) as the training objective function, which is a commonly used method of estimating the parameters of a probability distribution.

By the virtue of back-propagation algorithm, we can efficiently compute the gradient of L with respect to 0(theta) The training of NMT models usually adopts stochastic gradient search (SGD) algorithm. Instead of computing gradients on the full training set, SGD computes the loss function and gradients on a minibatch of the training set. The plain SGD optimizer updates the parameters of an NMT model. where α is the learning rate. With well-chosen learning rate, the parameters of NMT are guaranteed to converge into a local optima. In practice, instead of plain SGD optimizer, adaptive learning rate optimizers

Evolution of NMT:

Since 2013, there are attempts to build a pure neural MT. Early NMT architectures such as RCTM (Kalchbrenner and Blunsom, 2013), RNNEncdec and Seq2Seq (Sutskever et al., 2014) adopt a fixed-length approach, where the size of source representation is fixed regardless the length of source sentences. These works typically use recurrent neural networks (RNN) as the decoder network for generating variable-length translation. However, it is found that the performance of this approach degrades as the length of the input sentence increases (Cho et al., 2014b). Two explanations can account for this phenomenon: • 1.

The fixed-length representations have become the bottleneck during the encoding process for long sentences. As the encoder is forced to compress the entire source sentence into a set of fixed-length vectors, some important information may be lost in this process.

- 2.

  The longest path between the source words and target words , Sutskever et al. (2014) found that reverse the source sentence can significantly improve the performance of the fixed-length approach. By reversing the source sentence, the

paths between the beginning words of source and target sentences are reduced, thus the optimization problem becomes easier.

Due to these limitations, later NMT architectures switch to *variable-length* source representations, where the length of source representations depends on the length of the source sentence. The RNNsearch architecture *attention mechanism*, which is an important approach to implementing variable-length representations.

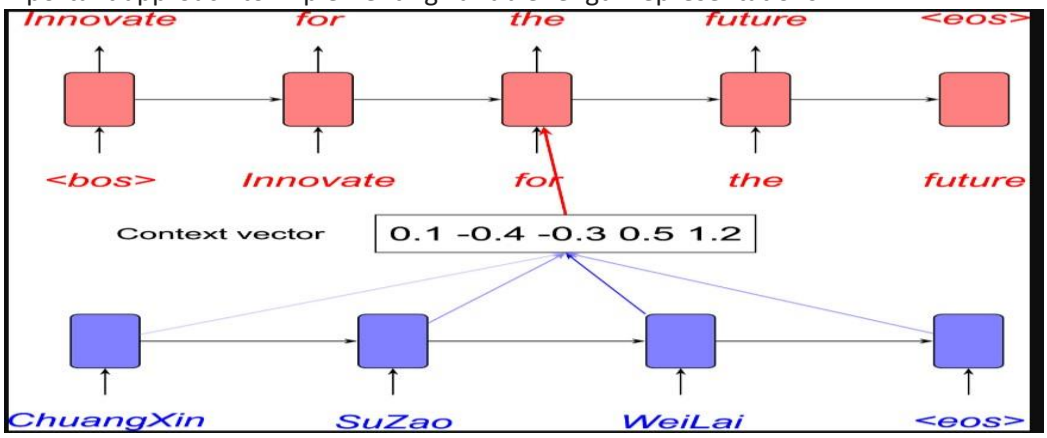


Figure 3: It shows the comparison between fixed-length and variable-length approaches. By using the attention mechanism, the paths between any source and target words are within a constant length. As a result, the attention mechanism has eased optimization difficulty.

Conclusion:

In conclusion, Neural Machine Translation (NMT) using the encoder-decoder architecture has proven to be a transformative approach to language translation, offering significant improvements over traditional methods. The encoder-decoder framework, often coupled with attention mechanisms, has become the cornerstone of state-of-the-art translation models, NMT has become the dominant approach to machine translation in both research and practice. This article reviewed the widely used methods in NMT, including modeling, decoding, data augmentation, interpretation, as well as evaluation. We then summarize the resources and tools that are useful for NMT research.

We list some important and challenging problems for NMT as follows:

Understanding NMT, Designing better architectures,
Making full use of monolingual data, Prior knowledge integration

References:

[1] Yuntian Deng, Anssi Kanervisto, Jeffrey Ling, and Alexander M. Rush. 2017. Image-to-markup generation with coarse-to-fine attention. In International Conference on Machine Learning (ICML).

[2] Hirschberg, J. & Manning, C. D. Advances in natural language processing. *Science* **349**, 261–266 (2015).

[3] Hajič, J. et al. *Natural Language Generation in the Context of Machine Translation* (Center for Language and Speech Processing, Johns Hopkins University, 2004).

[4] Bahdanau, D., Cho, K. & Bengio, Y. Neural machine translation by jointly learning to align and translate. In *International Conference on Learning Representations* (2014

[5] Hutchins, W. J. & Somers, H. L. *An introduction to machine translation*. (Academic Press, 1992).

[6] Bojar, O. et al. Findings of the 2018 conference on machine translation (WMT18). In *Proceedings of the Third Conference on Machine Translation (WMT)* 2, 272–307 (2018).

[7] Sennrich, R., Haddow, B. & Birch, A. Neural machine translation of rare words with subword units. *54th Annu. Meet. Assoc. Comput. Linguist*. https://doi.org/10.18653/v1/P16-1162 (2015).