A Descriptive Analysis of Statistical and Neural Machine Translation Techniques

1. Abstract

This paper presents a comparative analysis of SMT and NMT in terms of translation quality, computational efficiency, and usability. Recent studies comparing both approaches are reviewed to analyze their respective strengths and weaknesses. It is observed that NMT provides better translation quality than SMT, particularly for complex sentences and low-resource languages. However, SMT is still beneficial for languages with large parallel corpora and is computationally efficient. Neural Machine Translation (NMT) models are considered to be more reliable than Statistical Machine Translation (SMT) and other traditional forms of machine translation, particularly for under-resourced languages due to their self-learning capabilities. NMT models also offer faster translation speeds and higher quality outputs, which continue to improve over time as they learn from more data.

2. Introduction

Natural Language Processing (NLP)-based machine translation from any language to different language has gained significant attention in recent years. With the increase in globalisation, the demand for accurate and efficient translation has become more crucial than ever before. In this context, machine translation has emerged as a promising solution for cross-lingual communication.

It involves the use of algorithms and statistical models to automatically translate text from one language to another. This process involves several stages, including language identification, text pre-processing, alignment of source and target languages, and generation of translated text.

While machine translation has made significant strides in recent years, it still faces several challenges, such as handling idiomatic expressions, cultural nuances, and translating low-resource languages. Nevertheless, researchers continue to develop new approaches and techniques to improve the quality of machine translation.

In this research paper, we focus on the development of a machine translation system using NLP. Our goal is to evaluate the performance of different techniques and algorithms (i.e a Comparison between SMT and NMT) and identify the most effective approach for machine translation. This paper contributes to the growing body of research on machine translation and provides insights into the challenges and opportunities in this field.

3. Literature Review

[1] Ahmadnia, B., Dorr, B. J., & Kordjamshidi, P. (2020, September 27), it was found that In order to preserve source-language semantic relations in the corresponding target-language translation, neural machine translation (NMT) uses knowledge graphs (KGs). To improve source to target mapping, this is accomplished by embedding KG entity relations as embedding constraints.

There have been recent proposals for integrating external knowledge into neural machine translation (NMT) during training.[2] For example, Gulcehre et al. (2015) used monolingual data to train a neural language model, which was then integrated into the NMT decoder by concatenating the hidden states.[3] Arthur et al. (2016) biased the probability of the next target word in the NMT decoder using lexicon probabilities computed from a bilingual lexicon.[4] When linguistic information was available as external knowledge, Sennrich and Haddow (2015) computed separate embedding vectors for each aspect of linguistic information and concatenated them without altering the decoder.

To compare the performance of SMT and NMT systems, a study proposed development of an attention-based encoder-decoder neural network, which is described in detail below. The NMT model is based on the seq2seq model implemented by TensorFlow and adds features such as a bidirectional encoder, a beam-search decoder, a convolutional attention model, and a hierarchical encoder.

4. Approaches

Machine translation can be done in a variety of ways, In this paper, we are discussing only two of them including SMT, NMT and comparison between them showing which is the best approach :

- **SMT :**

  A computer translation system that was completely ignorant of linguistic principles and rules was first demonstrated at the IBM Research Center in early 1990. It examined comparable writings in two languages and looked for trends.

  How it works ?

  By analysing previously translated texts (also referred to as bilingual text corpora) and creating rules that are best adapted to translating a specific phrase, statistical machine translation (SMT) is carried out.

  More input in the necessary languages will increase the SMT's statistical likelihood of producing a translation that is more precise.

  Types of SMT :

  Word-based SMT :

  In Word-based SMT, the translation is done word by word without considering the context of the sentence. The translation model is based on a bilingual dictionary, which is used to translate each word from the source language to the target language. Word-based SMT does not take into account the grammar or syntax of the sentence.

  Example:

  English: I love pizza

Hindi:

Translation:

Syntax-based SMT :

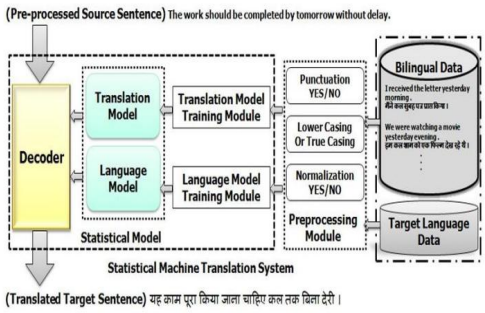
In Syntax-based SMT, the translation is done by considering the syntax or structure of the sentence. The translation model uses a parser to analyze the sentence and identify its grammatical structure, which is then used to generate the translated sentence in the target language. This approach is more effective than word-based SMT as it considers the context of the sentence.

Example:

English: She is going to the market.

Hindi:

Translation:



The Bayes' rule is a cornerstone of probability theory and is crucial to SMT.

Bayes' rule can be used in the setting of SMT to calculate the likelihood of a translation given a source sentence. To be more precise, we can calculate the posterior probability of a translation T given a source sentence S using Bayes' rule:

P(T|S) is equal to P(S|T)*P(T)/P.(S)

where P(T) is the prior probability of T, and P(S|T) is the probability of observing the source sentence S given the translation T.

For example, suppose we want to translate the sentence "Je suis heureux" from French to English. We have a bilingual corpus that contains translations of this sentence, and we want to use this corpus to estimate the probability of different translations.

Let T be the translation we want to estimate, and let S be the source sentence "Je suis heureux". We can use Bayes' rule to estimate the posterior probability of T given S:

P(T|S) = P(S|T) * P(T) / P(S)

To estimate the probability P(S|T), we can use a translation model that assigns a probability to each possible translation given the source sentence. For example, suppose the translation model assigns the following probabilities to three possible translations:

T1: "I am happy" with probability 0.8

T2: "I am glad" with probability 0.1

T3: "I am fortunate" with probability 0.1

Then we have:

P(S|T1) = 1 (since "Je suis heureux" is the source sentence for T1)

P(S|T2) = 0 (since "Je suis heureux" does not match "I am glad")

P(S|T3) = 0 (since "Je suis heureux" does not match "I am fortunate")

To estimate the prior probability P(T), we can use a language model that assigns a probability to each possible translation given the target language. For example, suppose the language model assigns the following probabilities to the same three possible translations:

T1: "I am happy" with probability 0.9

T2: "I am glad" with probability 0.05

T3: "I am fortunate" with probability 0.05

Then we have:

P(T1) = 0.9

P(T2) = 0.05

P(T3) = 0.05

To estimate the probability P(S), we need to compute the sum of the probabilities of observing S for all possible translations:

P(S) = Σ P(S|T) * P(T)

= P(S|T1) * P(T1) + P(S|T2) * P(T2) + P(S|T3) * P(T3)

= 1 * 0.9 + 0 * 0.05 + 0 * 0.05

= 0.9

Finally, we can compute the posterior

- **NMT:**

Neural Machine Translation is the most current method to machine translation, and it has shown a lot of potential in raising translation quality. The mapping between the source and target languages is learned by NMT using neural networks, a form of machine learning algorithm. NMT models can learn to catch intricate linguistic relationships and patterns because they are trained on huge parallel corpora.

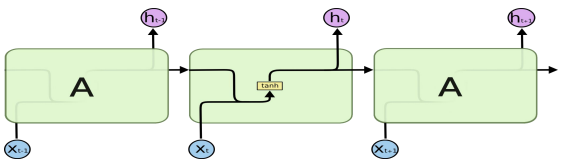
Neural Machine Translation (NMT) differs from traditional Phrase-Based Statistical Machine Translation (SMT) approaches that use separate subcomponents. The primary difference is that NMT uses vector representations, or "embeddings," for words and internal states, and simpler models compared to phrase-based models. There is no need for a separate language model, translation model, and reordering model; instead, NMT uses one word at a time is predicted by a single sequence model.
The complete source sentence and the entire already-written target sequence are, however, required for this sequence prediction to work.

To model word sequences, recurrent neural networks (RNNs) were initially used, with a bidirectional RNN (encoder) encoding the word prediction source for a second RNN (decoder) in the target language.

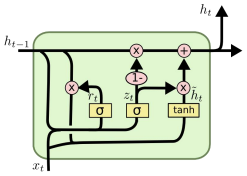
RNNs face difficulty in encoding long inputs, which can be compensated for by using an attention mechanism that allows the decoder to focus on different parts of the input while generating each word of the output. Several Coverage Models address issues with attention mechanisms, such as over- and under-translation due to ignoring past alignment information.

LSTM Networks for implementing seq2seq architecture of encoder-decoder

Long Short Term Memory Networks, more commonly referred to as "LSTMs," are a unique class of RNN that can recognise long-term relationships[5].They don't struggle to learn; rather, remembering knowledge for extended periods of time is practically their default behaviour. All recurrent neural networks take the shape of a series of neural network components that repeat. This recurring module in typical RNNs will be made up of just one tanh layer, for example.



The Gated Recurrent Unit, or GRU, developed by Cho, et al. is a striking modification to the LSTM. (2014) [5]. It creates a singular "update gate" by fusing the forget and input gates together. Along with making other adjustments, it combines the hidden state and cell state. The final model, which is more straightforward than traditional LSTM models, has been gaining popularity.



$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$
$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$
$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$
$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

Bidirectional LSTM:

If you comprehend LSTM, Bidirectional is quite easy to grasp. You can use a straightforward RNN (Recurrent Neural Network), GRU (Gated Recurrent Unit), or LSTM in a bidirectional network.(Long short Term Memory). In this essay, I'm going to use LSTM[6].

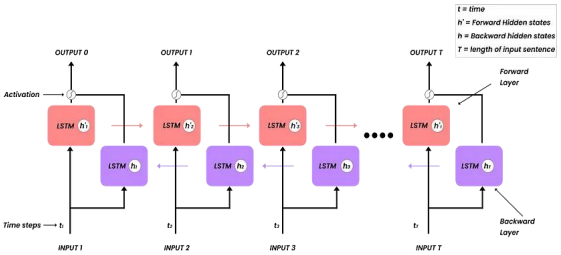
Our standard LSTM layer is the forward layer, but the backward LSTM layer has a flow that is in the opposite way.

Input is transmitted in both the forward and backward layers at every time step.

Each time step's outcome consists of both cells' output.(forward and backward layer). As a result, the prediction algorithm will also know the words that will come after.

Attention Mechanism:

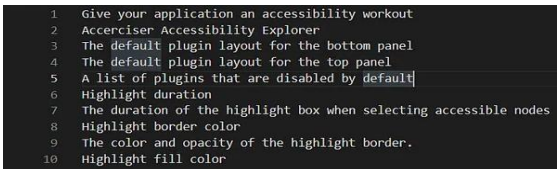
There are two key issues with the RNN-based seq2seq approach. Information loss and a vanishing gradient problem may occur because all the information cannot be compressed into a single fixed-size vector [7]. Consequently, accuracy suffers as input length increases. An attention mechanism has also been suggested as an emerging method for improving prediction accuracy, however this suffers from extended input sequences [8].In other words, the encoder's complete input sentence is used to anticipate the output word in every time step by the decoder. The part of the input word that is connected to the word to be predicted at that time receives more attention [9], despite the fact that not all input phrases are referred equally. Each input receives initial weights from the attention mechanism, which are modified throughout training based on the correlation between each input and the outcome prediction.

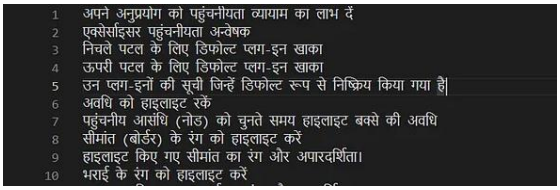


5. Methodologies

5.1 Dataset Used

We have used the " IIT Bombay English – Hindi Corpus " as it is one of the largest corpora currently accessible for translating between English and Hindi.





5.2 Preprocessing

As a first step in preprocessing, we must eliminate punctuation, numbers, and extra spaces before changing everything to lowercase [10].The prominence of the Hindi sentences is an essential point to keep in mind. The sentences must also be sanitised to eliminate any English characters before running them through the aforementioned function. In fact, this led to NaN bugs during training, which required a lot of debugging work. Regex can be used to solve the problem by substituting any English letter with an empty string.

LSTMs typically don't work well for lengthy sequences. As a result, we limit the number of sentences in our dataset to those that are no longer than 10 syllables. In order to reduce training time, we also take into account a total of 25000 sentences for example purposes [10].How will our algorithm know when to start and stop translating? For this, we use special words called "START" and "END" that we prepend and append to our target language.We still have character sentences as our data at this stage, which needs to be converted into numeric representations so the model can work with them. We must specifically build word-index representations for the Hindi and English sentences. We can use the built-in tokenizer in Tensorflow .

5.3 Set up of encoder and decoder

Encoder

As was said above, we'll utilise a bidirectional LSTM encoder to learn patterns in the input language (in this case, English). We'll use both the encoder's outputs and its states.(context vector[h, c])[11]. Bidirectional has both forward and backward modes, therefore taking both into consideration is a little different.(i.e. We will merge them.)

Decoder

Basic LSTM can predict the incorrect word occasionally because of word confusion.Therefore, if this kind of circumstance arises, the encoder step needs to look for the most pertinent information; this concept is known as "Attention."

In order for the decoder state to translate it better and the model to forecast it better, the encoder states read the sequential input and summarise it [11]. The summarised component then receives weights from the attention layer of the model.

5.4 Prediction

The attention model is used to predict output sequences. The encoder model learns features in input sentences and the decoder takes encoder states and predicts word by word using decoder inputs. The source and destination files are fed into the encoder layer after preprocessing. For training, the technique Stochastic Gradient Descent (SGD) was employed. Different combinations of encoder and decoder architectures were taught, including Deep Bi-LSTM, attention mechanisms, leftover connections, and a dense layer that serves as a link between the encoder and the decoder.

6. Conclusion-

There are more platforms and algorithms available for use with statistical machine translation (SMT) than there are with neural machine translation (NMT) because SMT has a longer history. As a result, SMT may be more accurate when translating and may be less expensive to implement and train as well as take up less virtual space.

SMT, however, is reliant on the calibre of the source material and necessitates bilingual text, which can be difficult for less widely spoken languages. However, because NMT's self-learning models can better account for context and provide more human-like translations, they can make it a more dependable solution, especially for under-resourced languages. NMT also has benefits in terms of speed and quality, which get better as the models keep learning.

NMT, the most recent development in machine translation, still has unrealized potential, while continual advancements in AI technology are accelerating development. However, human input is necessary for both SMT and NMT, especially during the early training stage. Both approaches require post-editing to guarantee the translated output satisfies the required quality criteria. Depending on the amount of editing necessary, post-editing is often divided into light and deep categories.