UNIVERSITY OF HERTFORDSHIRE

School of Computer Science

MSc Data Science and Analytics with Placement

7COM1075: Data Science and Analytics Masters Project

Date: 02/07/2021

Effectiveness of transformer model for machine translation compared to traditional encoder-decoder models

Name: Harsha Vardhan Bashetty

Student ID: 18055897

Supervised by: Thiago Matos Pinto

TABLE OF CONTENTS

1. Introduction and Overview

Deep learning enables feature extraction and transformation with the help of multiple layers of nonlinear processing units. In this class of machine learning algorithms, the output of each layer is used by the successive layer, learning multiple levels of representation. This representation corresponds to separate levels of abstraction (Deng and Yu, 2014).

Deep learning enables various capabilities such as computer vision with applications for image classification, image segmentation, object detection and tracking, Natural Language Processing (NLP) with applications for speech recognition, text classification, sentiment analysis and machine translation. These applications have use cases in many industries such as Agriculture, Aerospace, Defence, Finance, Healthcare, Manufacturing, supply chain, Retail and E-commerce (Takimoglu, 2021).

Machine translation investigates the use of computer software to translate text or speech from one language to another. Machine translation involves analyzing the structure of each phrase or term present in text or speech to be translated. This structure is used to break the elements that can be used to translate input language and recomposes the terms of the same structure in the output/target language (Madsen, 2009).

Human translators tend to take more amount of time for translating a lengthy article. This is because humans tend to forget the figures of speech used for describing certain words. They would require to constantly look up related references. While machine translation has the capacity to automatically translate and cam memorize key terms and phrases specific to any industry. This characteristic of machine translation leads to the very consistent translations generated across the complete document, which is highly arduous for multiple machine translators (Nirenburg, 1989).

1.1 Proposed method:

In this project, I am considering six encoder-decoder models: Uni-directional RNN, Bi-directional RNN, Uni-directional LSTM, Bi-directional LSTM, Attention and Transformer. The architecture for the encode-decoder model is described by Fig4 in the appendices section. These model architectures are built using python libraries TensorFlow (version 2.5) and Keras (version 2.5) and the performance is evaluated using Bilingual Evaluation Understudy Score (BLEU), which is used to compare the translation produced by the model to one or more reference translations. The flow chart describing the sequence of operations is shown in Fig1 in the appendices section.

1.2 Aim and Objectives:

1.2.1 Aim:

- Comparing the effectiveness of different encode-decoder models for machine translation using popular evaluation metric BLEU score

1.2.2 Objectives:

- Provide a brief overview of the progress made over the years in the field of NLP specific to machine translation
- Explain the recent breakthroughs in the field of machine translation like Attention and transformer models
- Study the effectiveness of the transformer models compared to much traditional seq2seq models

1.3 Libraries used:

Python programming language has a great source of libraries for building and implementing deep learning models. Its libraries TensorFlow and Keras are used to build, train and test the encoder-decoder models mentioned above. The python libraries NumPy, Pandas, Seaborn, Matplotlib and scikit-learn are used sparsely for data manipulation and visualization.

1.4 Ethical, legal, professional and social issues:

The opensource dataset contains the selected sentence pairs from the Tatoeba Project[1]. Tatoeba is a website where a user can translate the sentences provided by the other user. The users helped in translating will be acknowledged by the website. The sample of the dataset is shown by Fig3 in the appendices section.

The dataset taken from this project contains the usernames of the individuals who helped with translating specific sentences. For this project involving machine translation of sentences from one language to another, the username is not required and removed in the process of data cleaning. Thus, the models trained will be free from any kind of bias that may arise based on the personal information of an individual.

The project does not involve building applications that will be used by an end-user, hence there is no possibility of collecting personal information from the users.

I am aware of the fact that translations are generated by humans and possibly have an error and a model built based on the data could develop an underlying logic/structure based on

[1] Tatoeba website: https://tatoeba.org/en

these errors to generate new translations. Any such patterns/ behaviours will be identified and reported in detail.

The study does not involve any human participants and approval from University's Ethics Committee will not be required.

2. Progress to date

As of now, the detailed background study of relevant research papers is completed. For the implementation tasks, the finding of an appropriate dataset and the process of data preparation is completed.

2.1 Background study:

Usually, the typical neural machine translation models encode a source sentence into a fixed-length vector and using this vector, a decoder generates a translation. Bahdanau, Cho, and Bengio had discovered this fixed-length vector to be having a negative impact on translation quality, particularly for long sentences. They proposed a model architecture in which Recurrent Neural Network (RNN) focused on the relevant information for the generation of next target word only. This model had resulted in a great performance (Bahdanau, Bengio and Cho, 2016).

The attention mechanisms in machine translation allow models to focus on the selective parts of the encoder sentences. Two approaches: global and local, which looked at all source words and a subset of source words each time respectively are proposed. These model architectures are inspired by the attention mechanism. Both approaches were able to enhance the quality of translation on the WMT[2] translation tasks between English and German. The local attention approach, that focus on the subset of source words, had resulted in significant gains. Using an ensemble model, a new best result is established for WMT14 and WMT15 tasks (Luong, Pham and Manning, 2015).

The two machine translation models with different kinds of encoders: one with a gated recursive convolutional neural network and the other, an RNN with gated hidden units. They are able to produce accurate translations for short sentences without unknown words but suffered as sentences grew longer and with inclusion of the more unknown words (Cho, Merrienboer, Bahdanau and Bengio, 2014).

The GNMT (Google Neural Machine Translation) system, which is designed to work in the real world is able to produce translations indistinguishable from translations generated by humans. It designed to easily scale for applications in the real world. This is achieved by reducing the time for training and improving the speed of final translations. It is also able to achieve great success with rare words. For morphologically rich languages, the beam search technique is used, prompting the likeliness of covering all the available words in input/source sentences using ‘wordpiece’ modelling for the output sentences. This method resulted in about 60% in translation errors for simple sentences compared to the previous phrase-based system used by Google for production (Wu et al., 2016).

[2] WMT 2014 home page: http://www.statmt.org/wmt14/index.html

2.2 Finding dataset:

The translation task is carried out between English and French languages. The dataset contains about 190,000 sentences for French-English translations. This opensource dataset is found on the website[3] with the filename fra-eng.zip.

2.3 Data cleaning and preparation:

Firstly, the text containing undesirable characters is removed as the deep learning models will not be able to take input in the form of text. The input and target text has to be vectorized. The code used for this step is in Fig2 of the appendices section.

[3] Website: http://www.manythings.org/anki/

3. Planned work

	Task	Start Date	End Date	Duration (in days)	Status
1	Background study	09/06/2021	23/06/2021	14	Done
2	Finding Dataset	23/06/2021	30/06/2021	7	Done
3	Learning Python tools and libraries	15/06/2021	29/06/2021	14	Done
4	Data cleaning and preparation	30/06/2021	07/07/2021	7	Done
5	Implementing baseline models	07/07/2021	21/07/2021	14	Yet to be done
6	Model Tuning	21/07/2021	04/08/2021	14	Yet to be done
7	Model Evaluation	04/08/2021	11/08/2021	7	Yet to be done
8	Report Writing	11/08/2021	25/08/2021	14	Yet to be done

Table1: Describes each task for the project and its execution timeline

3.1 Pending works:

The table1 above clearly identify the individual tasks to be completed for finishing the project along with its Start date, End date, Duration and Status. As mentioned above the section, the first four tasks are finished. The other tasks explained below will be finished with in the durations mentioned in the table. The table is visually presented by the Gantt chart below.

3.1.1 Implementing baseline models:

Firstly, baseline models without any hyperparameter tuning will be done. This step will ensure the quick implementation of models to verify the model architecture and if any possible bottlenecks.

3.1.2 Model Tuning:

Each model will have a range of hyperparameters such as batch size, number of epochs, number of cells in encoder and decoder. After implementing the baseline model, each of the models is tuned to find appropriate values of hyperparameters to derive optimal performance. The various methods for regularization will be considered. All the changes made to the baseline model will be carefully documented and discussed in the final report.

3.1.3 Model Evaluation:

Each encoder-decoder model that is optimally tuned for its hyperparameter values is evaluated using the BLEU score. The results are compared and the reasons for varying levels of performance for each model will be presented.

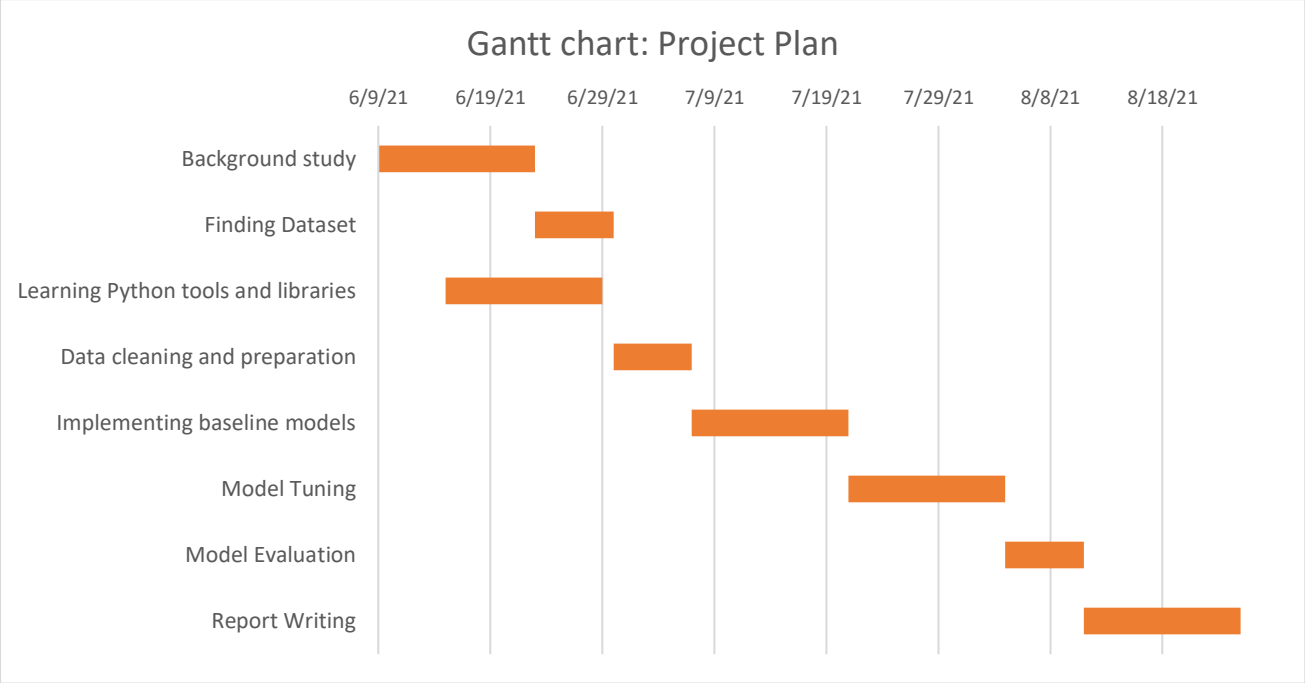

Fig: Gantt chart visually describing the tasks and their duration

4. Bibilography

- Bahdanau, D., Bengio, Y. and Cho, K., 2016. Neural Machine Translation by Jointly Learning to Align and Translate. [online] Available at: <https://arxiv.org/abs/1409.0473> [Accessed 12 June 2021].
- Cho, K., Merrienboer, B., Bahdanau, D. and Bengio, Y., 2014. On the Properties of Neural Machine Translation: Encoder-Decoder Approaches. [online] Available at: <https://arxiv.org/abs/1409.1259> [Accessed 12 June 2021].
- Deng, L. and Yu, D., 2014. *Deep learning*. 7th ed. Hanover, Mass.: Now Publishers, pp.197-387.
- Luong, M., Pham, H. and Manning, C., 2015. Effective Approaches to Attention-based Neural Machine Translation. [online] Available at: <https://arxiv.org/abs/1508.04025> [Accessed 13 June 2021].
- Madsen, M., 2009. The Limits of Machine Translation. [online] Available at: <https://www.semanticscholar.org/paper/The-Limits-of-Machine-Translation-Madsen/c3ec15cd591998821af5e731739083a5070ef063> [Accessed 11 June 2021].
- Nirenburg, S., 1989. *Knowledge-Based Machine Translation*. 4th ed. Springer, pp.5-24.
- Takimoglu, A., 2021. *Top 41 Deep Learning Use Cases/Applications/Examples in 2021*. [online] AIMultiple. Available at: <https://research.aimultiple.com/deep-learning-applications/> [Accessed 21 June 2021].
- Wu, Y., Schuster, M., Chen, Z., Le, Q., Norouzi, M., Macherey, W., Krikun, M., Cao, Y., Gao, Q., Macherey, K., Klingner, J., Shah, A., Johnson, M., Liu, X., Kaiser, Ł., Gouws, S., Kato, Y., Kudo, T., Kazawa, H., Stevens, K., Kurian, G., Patil, N., Wang, W., Young, C., Smith, J., Riesa, J., Rudnick, A., Vinyals, O., Corrado, G., Hughes, M. and Dean, J., 2016. Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation. [online] Available at: <https://arxiv.org/abs/1609.08144> [Accessed 21 June 2021].

5. Appendices:

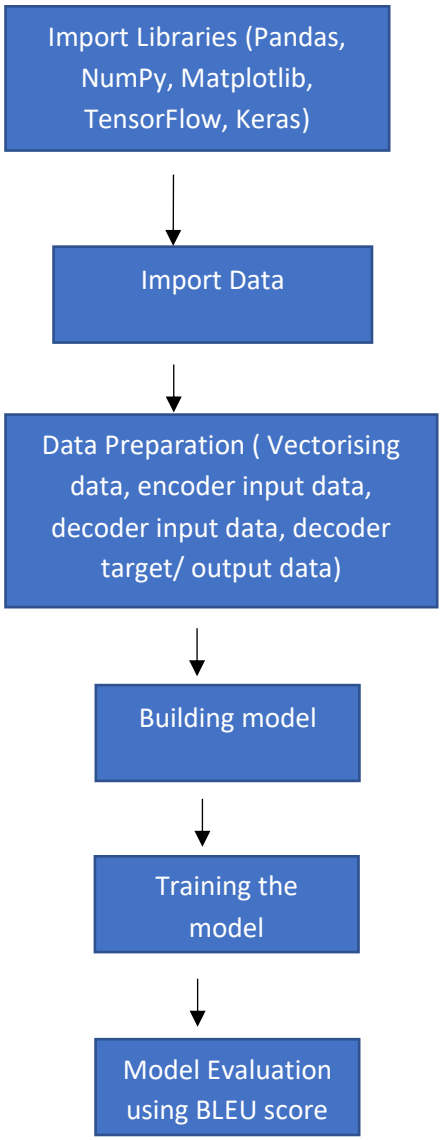

Fig1: Describe the sequence of steps to be followed for six encoder-decoder models

```
# Vectorize the data.
input_texts = []
target_texts = []
input_characters = set()
target_characters = set()
with open(data_path, "r", encoding="utf-8") as f:
    lines = f.read().split("\n")
for line in lines[: min(num_samples, len(lines) - 1)]:
    input_text, target_text, _ = line.split("\t")
    target_text = "\t" + target_text + "\n"
    input_texts.append(input_text)
    target_texts.append(target_text)
    for char in input_text:
        if char not in input_characters:
            input_characters.add(char)
    for char in target_text:
        if char not in target_characters:
            target_characters.add(char)

input_characters = sorted(list(input_characters))
target_characters = sorted(list(target_characters))
num_encoder_tokens = len(input_characters)
num_decoder_tokens = len(target_characters)
max_encoder_seq_length = max([len(txt) for txt in input_texts])
max_decoder_seq_length = max([len(txt) for txt in target_texts])

print("Number of samples:", len(input_texts))
print("Number of unique input tokens:", num_encoder_tokens)
print("Number of unique output tokens:", num_decoder_tokens)
print("Max sequence length for inputs:", max_encoder_seq_length)
print("Max sequence length for outputs:", max_decoder_seq_length)

input_token_index = dict([(char, i) for i, char in enumerate(input_characters)])
target_token_index = dict([(char, i) for i, char in enumerate(target_characters)])

encoder_input_data = np.zeros(
    (len(input_texts), max_encoder_seq_length, num_encoder_tokens), dtype="float32"
)
decoder_input_data = np.zeros(
    (len(input_texts), max_decoder_seq_length, num_decoder_tokens), dtype="float32"
)
decoder_target_data = np.zeros(
    (len(input_texts), max_decoder_seq_length, num_decoder_tokens), dtype="float32"
)

for i, (input_text, target_text) in enumerate(zip(input_texts, target_texts)):
    for t, char in enumerate(input_text):
        encoder_input_data[i, t, input_token_index[char]] = 1.0
    encoder_input_data[i, t + 1 :, input_token_index[" "]] = 1.0
    for t, char in enumerate(target_text):

        decoder_input_data[i, t, target_token_index[char]] = 1.0
        if t > 0:
            decoder_target_data[i, t - 1, target_token_index[char]] = 1.0
    decoder_input_data[i, t + 1 :, target_token_index[" "]] = 1.0
    decoder_target_data[i, t:, target_token_index[" "]] = 1.0
```

Fig2: show the code used for vectorizing the texts

```
Go.	Va !	CC-BY 2.0 (France) Attribution: tatoeba.org #2877272 (CM) & #1158250 (Wittydev)
Go.	Marche.	CC-BY 2.0 (France) Attribution: tatoeba.org #2877272 (CM) & #8090732 (Micsmithel)
Go.	Bouge !	CC-BY 2.0 (France) Attribution: tatoeba.org #2877272 (CM) & #9022935 (Micsmithel)
Hi.	Salut !	CC-BY 2.0 (France) Attribution: tatoeba.org #538123 (CM) & #509819 (Aiji)
Hi.	Salut.	CC-BY 2.0 (France) Attribution: tatoeba.org #538123 (CM) & #4320462 (gillux)
Run!	Cours !	CC-BY 2.0 (France) Attribution: tatoeba.org #906328 (papabear) & #906331 (sacredceltic)
Run!	Courez !	CC-BY 2.0 (France) Attribution: tatoeba.org #906328 (papabear) & #906332 (sacredceltic)
Run!	Prenez vos jambes à vos cous !	CC-BY 2.0 (France) Attribution: tatoeba.org #906328 (papabear) & #2077449
(sacredceltic)
Run!	File !	CC-BY 2.0 (France) Attribution: tatoeba.org #906328 (papabear) & #2077454 (sacredceltic)
Run!	Filez !	CC-BY 2.0 (France) Attribution: tatoeba.org #906328 (papabear) & #2077455 (sacredceltic)
Run!	Cours !	CC-BY 2.0 (France) Attribution: tatoeba.org #906328 (papabear) & #4580779 (franlexcois)
Run!	Fuyez !	CC-BY 2.0 (France) Attribution: tatoeba.org #906328 (papabear) & #7957917 (Micsmithel)
Run!	Fuyons !	CC-BY 2.0 (France) Attribution: tatoeba.org #906328 (papabear) & #7957918 (Micsmithel)
Run.	Cours !	CC-BY 2.0 (France) Attribution: tatoeba.org #4008918 (JSakuragi) & #906331 (sacredceltic)
Run.	Courez !	CC-BY 2.0 (France) Attribution: tatoeba.org #4008918 (JSakuragi) & #906332 (sacredceltic)
Run.	Prenez vos jambes à vos cous !	CC-BY 2.0 (France) Attribution: tatoeba.org #4008918 (JSakuragi) & #2077449
(sacredceltic)
```

Fig3: Showing sample of the considered dataset

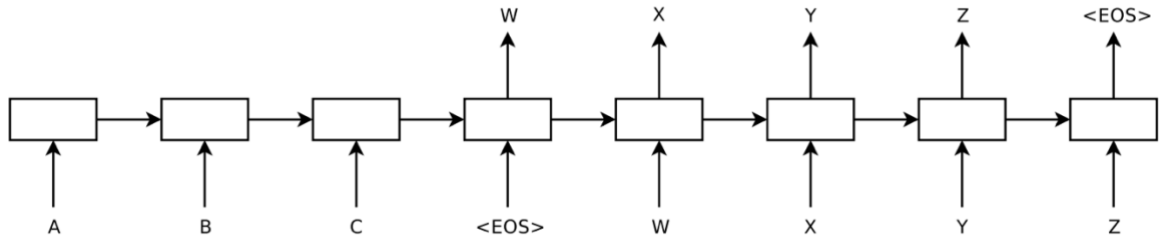

Fig4: Encoder-decoder model architecture