ON-DEVICE MACHINE TRANSLATION WITH REAL-TIME ELABORATION AND QnA

Aurosweta Mahapatra
206070948
aurosweta99@ucla.edu

Shrey Mathur
205928673
shreyzmail@gmail.com

Syam Sundar Kirubakaran
805905367
syamk@ucla.edu

Vaibhav Tiwari
105912719
vaibhavtiwari@ucla.edu

Abstract

This project addresses language barriers and translation latency by developing an on-device machine translation system for English to Spanish. It employs RNN and Transformer models, evaluating translation quality using BLEU and Rouge metrics. The best model, the RNN, is converted to TensorFlow Lite for practical deployment. The system incorporates on-device text recognition from still images, allowing instant English-to-Spanish translations. Its unique feature is real-time elaboration, providing additional contextual information beyond translation. This enhances user experience and enables seamless communication across languages.

1. Introduction

1.1 Background

Language barriers and translation latency are persistent challenges in effective global communication. To address these issues, this project focuses on developing an on-device machine translation system that offers real-time elaboration and question-and-answer (QnA) capabilities. Our specific translation task involves converting English text to Spanish, employing recurrent neural network (RNN) and Transformer models. Evaluation of translation quality utilizes standard metrics like BLEU and Rouge.

To ensure practical implementation, we convert our best-performing model, the RNN, from TensorFlow to TensorFlow Lite, optimizing it for deployment on edge devices. We also integrate on-device text recognition for still images, allowing instant English-to-Spanish translations.

The unique feature of our system lies in its real-time elaboration, going beyond basic translation. For example, if a user inputs a question such as "How to make an omelette," our system not only provides the translation but also offers step-by-step instructions in Spanish. This anticipates users' potential future queries, enhancing the user experience.

1.2 Dataset

We have taken the dataset from the sentences_detailed.csv file from tatoeba.org. (http://tatoeba.org/files/downloads/sentences_detailed.csv).

2 Models

This project delves into the exploration of two different models for our machine translation task: RNN and Transformer. The objective is to evaluate their performance and effectiveness in addressing these issues. In the following sections, we provide detailed explanations of each model and their respective methodologies.

2.1 RNN

A Recurrent Neural Network (RNN) is a type of neural network that is well-suited for processing sequential data. It's architecture allows it to retain information from previous steps and utilize it in subsequent steps, allowing it to capture temporal dependencies. Since RNNs can process sequential data and capture contextual dependencies between words, Neural machine translation systems are typically implemented with a Recurrent Neural Network (RNN) based encoder-decoder framework (Bahdanau et al., 2016). For our experiment, we decided to use a bidirectional RNN encoder-decoder architecture (Bahdanau et al., 2016; Yang et al., 2017). In this setup, the source language sentence is encoded using a bidirectional RNN. The encoded representation of the source sentence is then passed to a decoder RNN, which generates the corresponding translated sentence in the target language. We have kept dimensionality of the embedded layer as 256, batch size as 64 and used Adam optimizer (Kingma and Ba, 2014). Due to limited compute, we have used only 1 encoder and decoder layer. Following (Srivastava, 2013), we used dropout after the RNN decoder layer. By considering past and future contexts of each word, bidirectional RNNs are able to better capture contextual information, leading to more accurate translations, as compared to their vanilla counterpart. This was the motivation behind using bidirectional RNN.

Since RNNs process data in a sequential manner, they can be slow. Another drawback of RNN is that it is not able to retain long term dependencies. So, another popular choice can be to use RNN with Long-Short Term Memory (LSTM) for the machine translation tasks (Jozefowicz et al.,2016; Lample et al.,2018). However, our dataset didn't comprise of very long sentences, and due to limited computing power, we decided not to use LSTM.

2.2 Transformer

A transformer is a neural network architecture that uses attention mechanism to process sequential data. Unlike recurrent neural networks (RNNs), transformers operate in parallel, enabling more efficient computation. We are chosing the Transformer model (Vaswani et al., 2017) as it has shown to be very effective for Machine Translation tasks (Ott et al., 2018), including multilingual machine translation tasks (Lakew et al., 2018; Sachan and Neubig, 2018). This is due to their attention mechanism, parallel processing, and encoder-decoder architecture. The attention mechanism allows them to capture dependencies between words in a sentence, facilitating the understanding of contextual relationships. Parallel processing enables efficient handling of long sequences, and the encoder decoder architecture handles variable-length input and output sequences.

For our experiment, we focused on the Transformer in the "Base" configuration. We refer the reader to Vaswani et al. (2017) to get a better understanding of the model architecture. Due to compute restrictions, we have only used 1 encoder and 1 decoder block. The encoder comprises of 8 attention heads, 1 attention layer, 2 add and normalization layers and 2 fully connected layers with 2048 and 256 neurons. The decoder block comprises of 8 attention heads, 2 attention layers (self and cross attention). 3 add and normalization layers and 2 fully connected layers with 2048 and 256 neurons. We have kept dimensionality of the embedded layer as 256, batch size as 64 and used Adam optimizer (Kingma and Ba, 2014)

3 Workflow

The workflow of our project is mentioned in this section with a detailed diagram in figure 1.

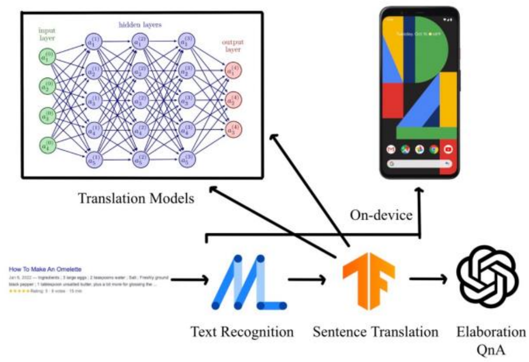

Figure 1: Project workflow

After training the selected model for an adequate number of epochs, the TensorFlow model is converted into a TensorFlow Lite model using the following command:

```
tflite_convert --saved_model_dir=
/path/to/saved_model --output_file=
/path/to/output.tflite
```

The primary purpose of this conversion is to leverage the advantages of TensorFlow Lite models. Unlike TensorFlow models, these lightweight models allow us to deploy high-performance models on embedded systems or mobile devices without significant degradation in performance or accuracy. An additional advantage is the availability of prebuilt and popular Android SDKs, such as MLKit, which natively support the loading of these lite models at runtime on any Android device. The converted model is then saved in the Android resource directory and loaded at runtime using the following code:

```
val localModel = LocalModel.Builder()
                    .setAssetFilePath(MODEL)
                    .build()
```

In our implementation, we utilize two models: MLKit's native OCR model for text recognition and a custom-built translation model developed by our team. The Optical Character Recognition model is imported as a vision dependency using "com.google.ml.vision.DEPENDENCIES". We process the bitmap image through the OCR model and utilize convenient methods such as "onSuccessListener" and "onFailureListener" to obtain the desired results. The recognized text is extracted from the onSuccess method and passed into the TFLite translation model, which is loaded using MLKit. It is worth mentioning that the OCR model is downloaded during the initial run of the app and cached for subsequent runs. However, this may lead to race conditions if users attempt to translate text before the OCR model finishes loading.

Once the translation model is loaded, we set the source and target languages as translator options and invoke the on-device model to obtain the translated text:

```
val options = TranslatorOptions.Builder()
            .setSourceLanguage(TranslateLanguage.ENGLISH)
            .setTargetLanguage(TranslateLanguage.SPANISH)
            .build()
```

In this specific scenario, we set the source language to English and the target language to Spanish. The translated text is obtained through the "onSuccess" method, similar to the callback methods used for the OCR model. We display the recognized and translated text on the user interface as soon as we receive it from the model.

To provide further elaboration of the translated text, we make a POST call to OpenAI's completion API (https://api.openai.com/v1/chat/completions). It is important to obtain a bearer token from OpenAI's developer dashboard for authenticating the API calls.

Figure 2 displays the translation result: Notice how the language translation model gives the correct result despite a word gap issue with the recognized text.

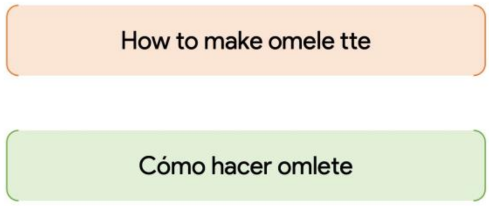

Figure 2: Translation Result

4 Result and Comparison

Initially, we used Precision, Recall and F-1 score to evaluate our models. However, we were not getting satisfactory results. This is expected because these metrics on their own are not able to capture nuanced nature of translation quality. Precision and recall are metrics commonly used in tasks like information retrieval or binary classification, where the goal is to classify instances into categories. However, in machine translation, the output is a sequence of words or phrases, and a direct classification evaluation is not appropriate. F-1 score combines precision and recall into a single metric, which can be useful in certain tasks. However, it would still not be able to capture the complexities and subtleties of translation quality. So, we decided to use BLEU (Bilingual Evaluation Understudy) and ROUGE (RecallOriented Understudy for Gisting Evaluation) metrics as these will consider factors such as n-gram overlap and semantic similarity providing a more comprehensive assessment of translation quality. As we can see from the below table Bidirectional RNN Encoder Decoder outperforms Transformer model the main reason behind it is that the transformer model has just one encoder-decoder block and has been trained for insufficient number of epochs due to compute constraints.

Following are the BLEU and ROUGE scores for English to Spanish translation task.

	BLEU	ROUGE -1	ROUGE - 2	ROUGE-L
Bidirectional RNN	0.619	0.742	0.503	0.729
Transformer	0.571	0.681	0.409	0.67

Table 1: Comparing evaluation metrics

Futhermore, we also successfully ported it on-device and tested the performance of our model on android, the screenshots attached below testify our successful implementation of the project.

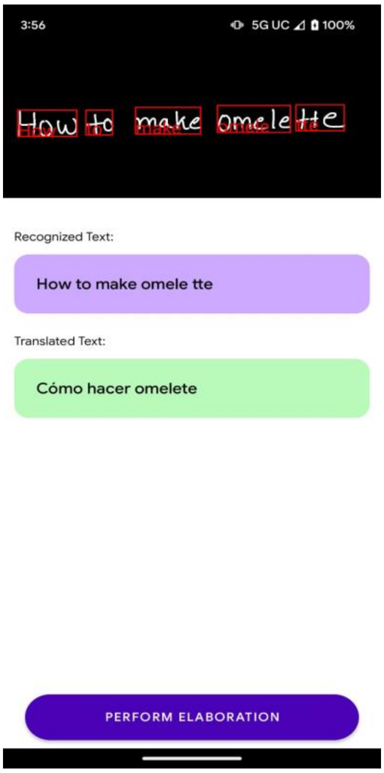

Figure 3: Translation Result- On-device

Figure 3 shows the result of processing and extracting the text from the image using Optical Character Recognition(OCR) model and translation results using our custom model. Both of them uses MLKit as SDK on android.

Also, Figure 4 below shows the results of the remote API call made to OpenAI services using the "gpt-3.5-turbo" model that gives the context aware elaboration results.

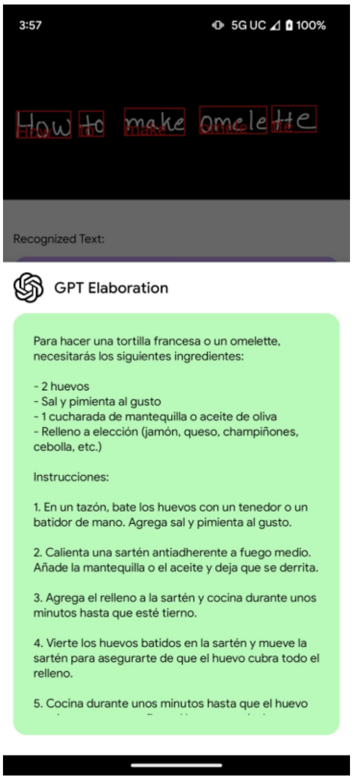

Figure 4: Elaboration Result- On-device

5 Conclusion

In conclusion, this project successfully developed a machine translation system that operates on user devices and offers real-time elaboration and Q&A capabilities. The RNN model demonstrated superior performance compared to the Transformer model, achieving notable BLEU (0.617) and Rouge scores across different metrics. The utilization of TensorFlow Lite enabled efficient deployment on edge devices, while on-device text recognition facilitated instant translations from English to Spanish. The incorporation of real-time elaboration enhanced the user experience, and the integration of OpenAI's completion API provided additional information. Overall, this project significantly contributes to overcoming language barriers, with potential avenues for further exploration and optimization of models in the future.

6.Acknowledgments

We would like to thank Prof. Kai Wei Chan for encouraging us to try new cutting edge projects and helping us throughout the coursework. We also like to thank the course TAs Tanmay, Elaine and Masoud for helping us in every stage of the project.

References

1. D. Bahdanau, K. Cho, and Y. Bengio. Neural Machine Translation by Jointly Learning to Align and Translate. 2016. arXiv preprint arXiv:1409.0473 [cs.CL].

2. Rafal Jozefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, and Yonghui Wu. 2016. Exploring the Limits of Language Modeling. In Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP).

3. D. P. Kingma and J. Ba. "Adam: A method for stochastic optimization." 2014. arXiv preprint arXiv:1412.6980.

4. Surafel Melaku Lakew, Mauro Cettolo, and Marcello Federico. 2018. A Comparison of Transformer and Recurrent Neural Networks on Multilingual Neural Machine Translation. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 641–652, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

5. Guillaume Lample, Myle Ott, Alexis Conneau, Ludovic Denoyer, and Marc'Aurelio Ranzato. 2018. Phrase-Based & Neural Unsupervised Machine Translation. In Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP).

6. Myle Ott, Sergey Edunov, David Grangier, and Michael Auli. 2018. Scaling Neural Machine Translation. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 1–9, Brussels, Belgium. Association for Computational Linguistics.

7. Devendra Sachan and Graham Neubig. 2018. Parameter Sharing Methods for Multilingual Self-Attentional Translation Models. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 261–271, Brussels, Belgium. Association for Computational Linguistics.

8. Nitish, Srivastava. 2013. Improving neural networks with dropout. PhD thesis, University of Toronto.

9. Zhen Yang, Wei Chen, Feng Wang, and Bo Xu. 2017. Improving Neural Machine Translation with Conditional Sequence Generative Adversarial Nets. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1346–1355, New Orleans, Louisiana. Association for Computational Linguistics.

10. A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. "Attention Is All You Need." 2017. arXiv preprint arXiv:1706.03762 [cs.CL].