NEURAL LANGUAGE TRANSLATION

LAUREN O'DONNELL

DS 6050 DEEP LEARNING

AGENDA

- Introduction
- Data
- Methods
- Results
- Conclusions

INTRODUCTION

- Neural Machine Translation (NMT) is a branch of Machine Translation (MT) which falls under Natural Language Processing (NLP)
    - Aims to translate natural language sentences
- Used in Google Translate, Amazon Translate, and many other common translation services & applications
- Many use cases: book translations, caption translation, research paper/website translations, etc.
- Consistency is an outstanding challenge!

INTRODUCTION: A BRIEF HISTORY

- Early approaches relied on hand-crafted translation rules and linguistic knowledge
  - Challenges: language is COMPLEX and there are many language IRREGULARITIES
- Statistical Machine Translation (SMT)
  - Learns latent structures
  - Uses discrete symbolic representations
  - Separately tuned components
  - Challenges: unable to model long-distance dependencies between words; poor quality
- Neural Machine Translation (NMT)
  - Single large neural network to model entire translation process
  - Uses continuous symbolic representations
  - End-to-end training
  - Probabilistic framework

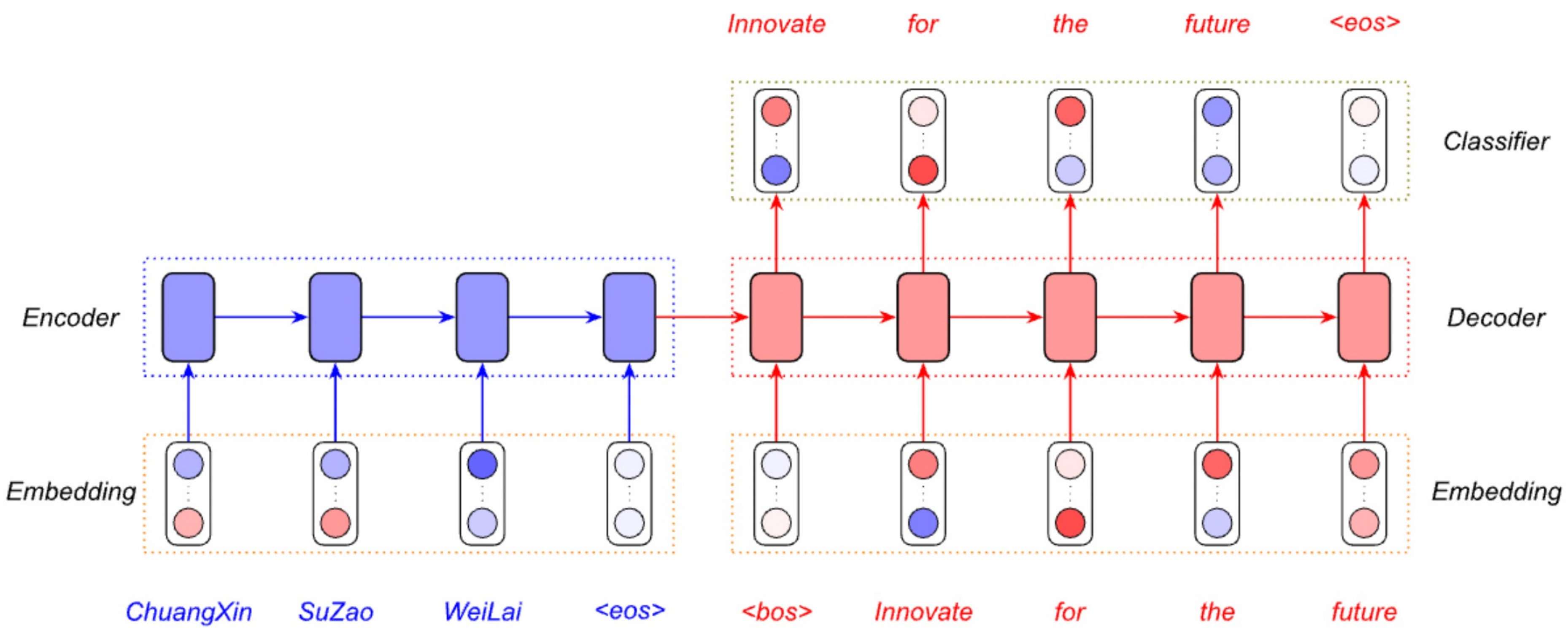
INTRODUCTION:
NMT ARCHITECTURE

- NMT is a sequence-to-sequence model
  - Given a source sentence $x = \{x_1, \dots, x_S\}$ and a target sentence $y = \{y_1, \dots, y_T\}$, the chain rule and the conditional distribution can factorize from left-to-right (L2R) as:

$$P(y|x) = \prod_{t=1}^{T} P(y_t | y_0, \dots, y_{t-1}, x)$$

- Three main sections:
  - Embedding layers
  - Encoder and decoder networks
  - Classification layer
- Common encoders and decoders include RNN, CNN, and SAN

INTRODUCTION:
NMT ARCHITECTURE



INTRODUCTION: ENCODERS & DECODERS

RECURRENT NEURAL NETWORKS

- Pros
  - Extremely powerful
  - Infinite receptive fields
  - Does not require additional padding or masking due to sequential nature
- Cons
  - Not compatible with GPU/TPU
  - Severe vanishing and exploding gradient problems

CONVOLUTIONAL NEURAL NETWORKS

- Pros
  - Compatible with GPU/TPU
- Cons
  - Requires additional padding and masking to prevent the network from seeing future words
  - Limited receptive field

SELF ATTENTION NETWORKS

- Pros
  - Compatible with GPU/TPU
  - Infinite receptive fields
- Cons
  - Requires additional padding and masking to prevent the network from seeing future words

INTRODUCTION: GOALS

- Explore and compare two techniques and architectures for NMT:
  - RNN and SAN

DATA

- English-Portuguese datasets
- Contains 168,903 total records
  - Mix of words (like exclamations) and sentences
  - 80-20 training to validation set split
- Features are the words!
  - Varies sentence to sentence and phrase to phrase
- Kaggle Notebook pre-processed the data and generated an RNN model utilizing GRU and Attention Mechanism

METHODS: FUNDAMENTAL ARCHITECTURES

RNN: GATED RECURRENT UNIT (GRU) WITH ATTENTION MECHANISM

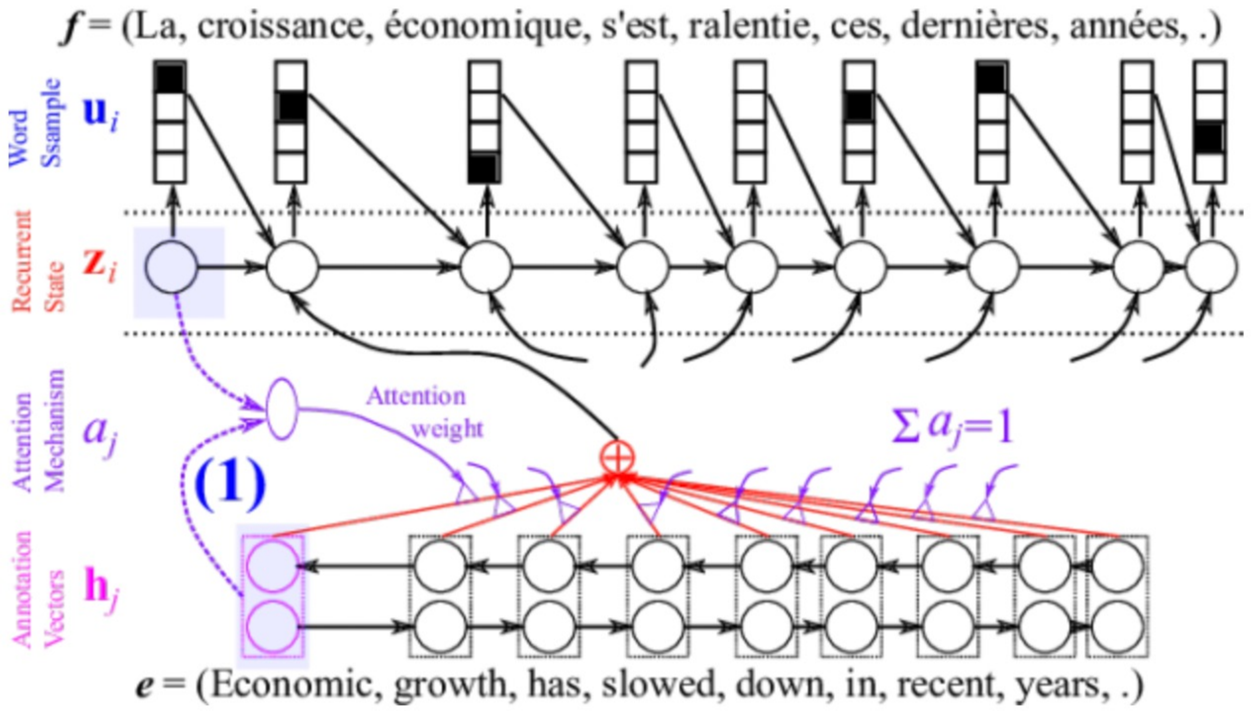
- RNN model
- Basic encoder-decoder
- Consists of 1 encoder GRU layer and 1 GRU decoder layer with an attention mechanism

SAN: BIDIRECTIONAL ENCODER REPRESENTATIONS FROM TRANSFORMERS (BERT)

- SAN model
- Additional sinusoid-style position encoding added to input embedding
- Consists of 6 encoder layers and 6 decoder layers

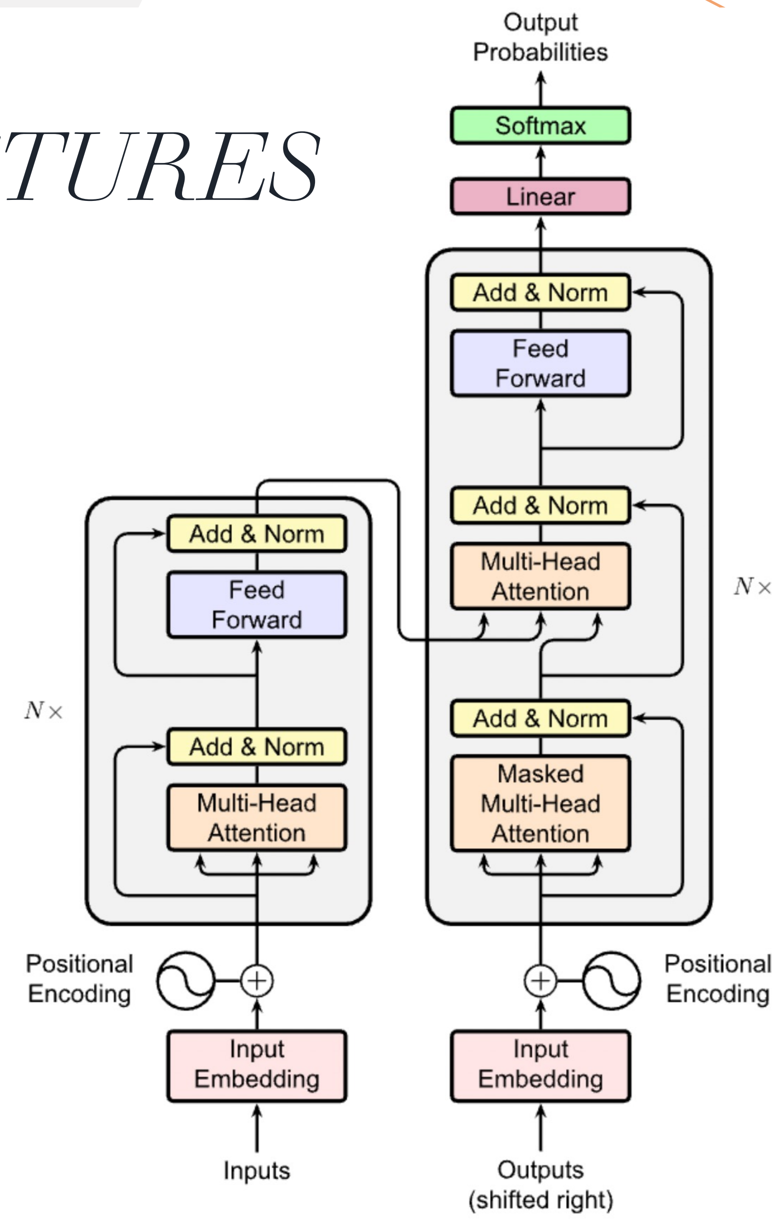
METHODS:
FUNDAMENTAL ARCHITECTURES

RNN: GATED RECURRENT UNIT (GRU) WITH ATTENTION MECHANISM



METHODS: FUNDAMENTAL ARCHITECTURES

TRANSFORMERS



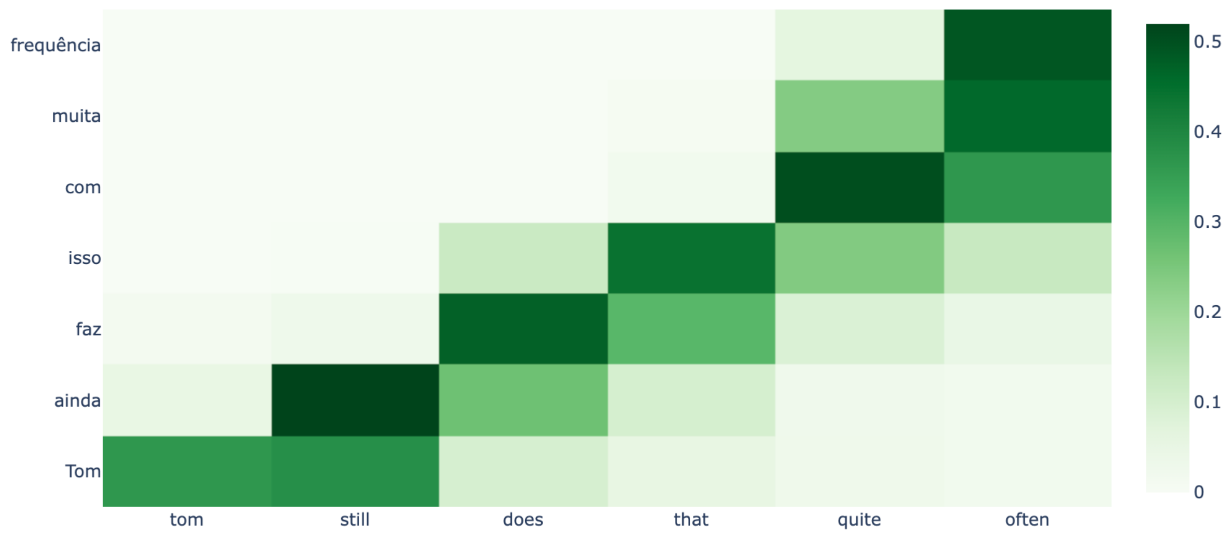
METHODS: HYPERPARAMETERS

- Learning Rate: 0. 000001
- EPOCHS: 10
- Batch Size: 100
- Activation Function: softmax for RNN, sigmoid for SAN
- Dropout Rate: 0.1

```
class Encoder(tf.keras.Model):
    def __init__(self, vocab_size, embedding_dim, enc_units, batch_sz):
        super(Encoder, self).__init__()
        self.batch_sz = batch_sz
        self.enc_units = enc_units
        self.embedding = tf.keras.layers.Embedding(vocab_size, embedding_dim)
        self.gru = gru(self.enc_units)

    def call(self, x, hidden):
        x = self.embedding(x)
        output, state = self.gru(x, initial_state = hidden)
        return output, state

    def initialize_hidden_state(self):
        return tf.zeros((self.batch_sz, self.enc_units))
```

```
class BERT_Encoder(tf.keras.Model):
    def __init__(self, max_length, num_classes):
        super(BERT_Encoder, self).__init__()
        self.max_length = max_length
        self.num_classes = num_classes
        self.bert = TFBertModel.from_pretrained('bert-base-uncased')
        self.dropout = tf.keras.layers.Dropout(0.1)
        self.classifier = tf.keras.layers.Dense(num_classes, activation='sigmoid')

    def call(self, inputs, **kwargs):
        _, pooler_output = self.bert(inputs, **kwargs)
        x = self.dropout(pooler_output, training=kwargs.get('training', False))
        x = self.classifier(x)
        return x
```

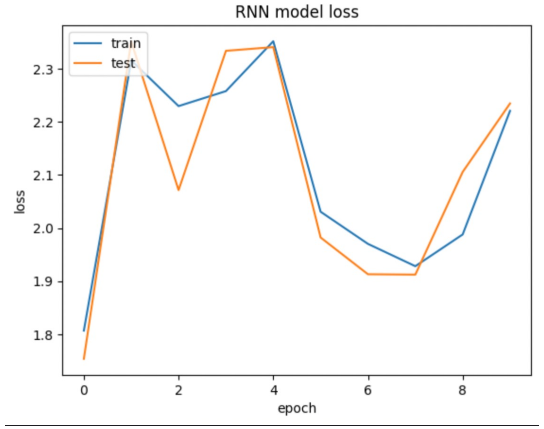
```
Input: tom still does that quite often
Predicted translation: Tom ainda faz isso com muita frequência
Actual translation: Tom ainda faz isso com bastante frequência
```

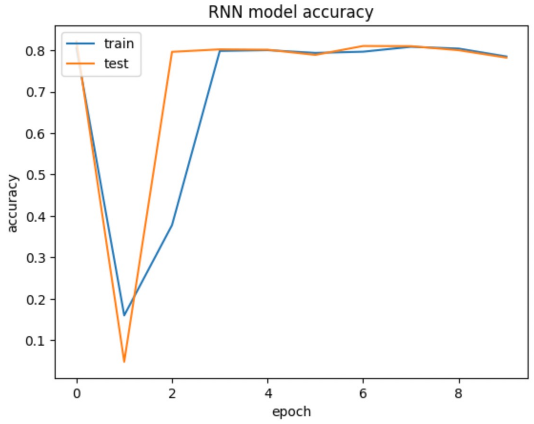


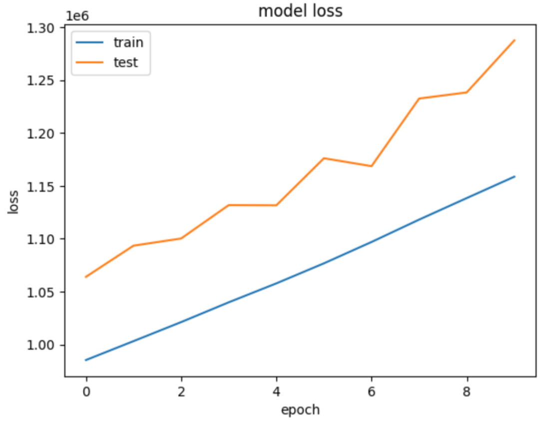
RESULTS

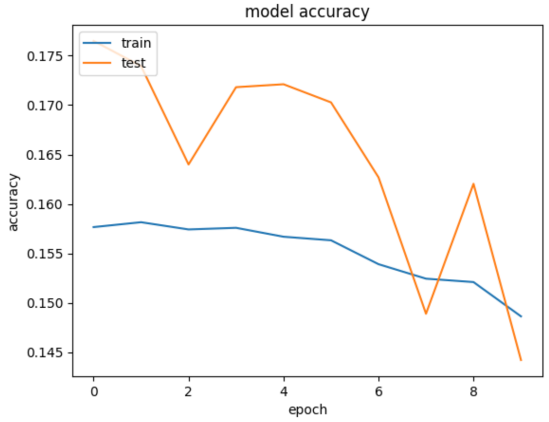
- RNN model:
  - Training accuracy: 78.48%
  - Validation accuracy: 78.20%

- BERT model:
  - Training accuracy: 14.86%
  - Validation accuracy: 14.42%


RNN model loss
train
test
loss
epoch
2.3
2.2
2.1
2.0
1.9
1.8
0
2
4
6
8


RNN model accuracy
train
test
accuracy
epoch
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
2
4
6
8


model loss
1e6
train
test
loss
epoch
1.30
1.25
1.20
1.15
1.10
1.05
1.00
0
2
4
6
8


model accuracy
train
test
accuracy
epoch
0.175
0.170
0.165
0.160
0.155
0.150
0.145
0
2
4
6
8

RESULTS

CONCLUSIONS

RESEARCH CONCLUSIONS

- RNN model performed better
  - Training accuracy: 78.48%
  - Validation accuracy: 78.20%
- Language translation is COMPLEX
  - Easy to overfit

FUTURE RESEARCH & RECOMMENDATIONS

- Significantly more compute resources are required
- Requires more training time
- How do other languages either more similar or different affect results? Ex.:
  - Spanish to Portuguese
  - English to Arabic

REFERENCES

- Zhixing Tan, Shuo Wang, Zonghan Yang, Gang Chen, Xuancheng Huang, Maosong Sun, and Yang Liu. Neural machine translation: A review of methods, resources, and tools. AI Open, 1:5–21, 2020.
- Kalchbrenner, Nal and Espeholt, Lasse and Simonyan, Karen and Oord, Aaron van den and Graves, Alex and Kavukcuoglu, Koray. Neural machine translation in linear time. arXiv preprint arXiv:1610.10099, 2016.
- *Manning Luong, Pham. Effective approaches to attention-based neural machine translation. https://paperswithcode.com/paper/effective-approaches-to-attention-based, 2015.*
- A history of machine translation from the Cold War to deep learning. Freecodecamp, https://www.freecodecamp.org/news/a-history-of-machine-translation-from-the-cold-war-to-deep-learning-f1d335ce8b5/#:~:text=On%20January%207th%201954%2C%20at,the%20first%20time%20in%20history, 2018.
- Rani Horev. BERT Explained: State of the art language model for NLP. Towards data science, https://towardsdatascience.com/bert-explained-state-of-the-art-language-model-for-nlp-f8b21a9b6270, 2018.
- https://huggingface.co/
- https://www.kaggle.com/code/walterfly/use-hugging-face-s-tensorflow-2-transformer-models
- https://www.tensorflow.org/text/tutorials/nmt_with_attention
- https://www.tensorflow.org/text/tutorials/transformer
- https://www.kaggle.com/code/nageshsingh/neural-machine-translation-attention-mechanism/notebook