# Development of Neural Machine Translation for Healthcare Communication/Translation

| | |
|---|---|
| **Summary** | Neural Machine translation for Biomedical Data |
| **Document URL** | https://github.com/ziqlu0722/INFO7374-FINAL-MEDICAL-NMT |
| **Application URL** | https://tinyurl.com/info7374team2 |
| **Author** | Group 2: Vignesh Murali, Ziqing Lu, Ritvik Reddy |

---


## Abstract

The quality of machine translation is rapidly evolving. In this project, we aim to build an end to end version of neural networks starting from the code in tensorflow to running the model in the cloud which can maximize the translation performance. We implemented sequence to sequence approach which generally belongs encoder-decoder in which source sentence is a vector of inputs (embeddings) which is in turn decoded into target translation. In our project, we examine a ton of architectural combinations for translating from English to French by using a large biomedical corpus. The comparison of different architectures and then implementing a real-time translator using attention layer to preserve the semantics of input language is the primary focus of this design of experiments.


---

## Introduction

Neural models involve building an end-to-end neural network that maps aligned bilingual texts which, given an input sentence X to be translated, is normally trained to maximise the probability of a target sequence Y without additional external linguistic information.

Recently, a surge of interest in NMT came with the application of deep neural networks (DNNs) to build end-to-end encoder-decoder models. Bahdanau first introduced an attention mechanism into the NMT encoder-decoder framework which is trained to attend to the relevant

source-language words as it generates each word of the target sentence. Some important recent developments in NMT involve improving the attention mechanism, including linguistic information or including more languages into the model

### Overview Of  Neural Machine Translation in Healthcare

Making medical information understandable is relevant to both physicians and patients. As an example, Healthcare Technologies emphasizes that a foreign patient may need a description and explanation of their diagnosis, along with a related and comprehensive set of information. In most countries, residents and immigrants communicate in languages other than the official one.

Google Translate has lower accuracy when used for medical phrase translations and should not be trusted for important medical communications. However, it still remains the most easily available and free initial mode of communication between a doctor and patient when language is a barrier.

Hence, we are proposing an NMT system which aims to provide better translations in the medical domain.

### Goals

To Improve the Neural Machine Translations for Medical Domain thereby increasing physician-patient engagement and hence enhance the total quality of care provided to the patients.

To Tabulate various Machine Translating sequence to sequence approaches and determine which architecture shows evidence of promise for smaller data and then train that model/architecture for a bigger corpora using AWS Cloud platform and then deploy it.

---

## Data and Data Preprocessing

Experiments were performed by using the Biomedical Domain Dataset. It contains a huge parallel corpus of  Eng - French Translations. Dataset used in the experiment is from [WMT Biomedical Translation Task](). In total there are more than 600K English-French sentence Pairs. Some details are given below:

- Language: English and French Sentence Pairs
- Number of Sentence Pairs: 612,878
- Vocabulary Size: 118,476 (English), 151,377 (French)

Due to resource and time limitation, most of our experiments are based on 50K and

100K sentence pairs from this dataset.

Our data consisted of 612,878 sentences which had both English and its French translations. The data is of healthcare domain which mainly consists of medical products, diagnostics, pharmacology and patient-physician interactions.

The data was read using UTF-8 encoded text format and then stripped in new lines and appended them together.First data was split into train and test and data was normalized and converted into lower case and tokenized along with padding the sentences with <sos> start of sentence <eos> end of the sentence and <unk> for unknown sentence.

---

# Modelling and Implementation

## 1.Training Setup

- Training Platform: google Colab GPU, and AWS EC2 (p3.2xLarge 1 GPU with 8 CPU GPU Memory=16 GB storage space=250GB)
- Programming language: Python.
- Modelling Framework: Tensorflow Backend
- Evaluation: mean BLEU, Loss, Seconds/Epoch.
  We only limit most of the experiments to 20 epochs due to time and computation power limitation
- Experiment Scope: 1~5 layers of Encoder and Decoder, Attention Mechanism(Bahanadau,Luong), 128 to 512 number of units, different cell-types(LSTM, Bidirectional LSTM, GRU, Bi-Directional GRU)
- Basic Structure: Seq2Seq with Attention Mechanism

## 2.Software

- Web application: Flask
- Final Deployed Model is trained on AWS EC2
- Modelling Framework: Tensorflow
- Programming Language: Python

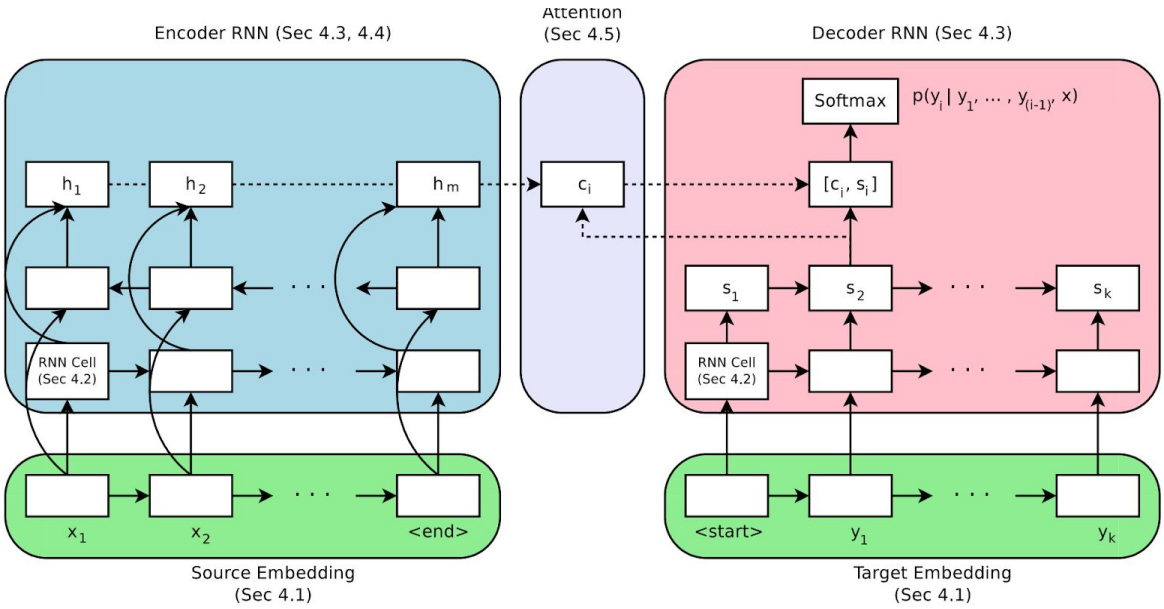



Figure 1: This figure is from the paper: *Massive Exploration of Neural Machine Translation Architectures*

## Pipeline Design

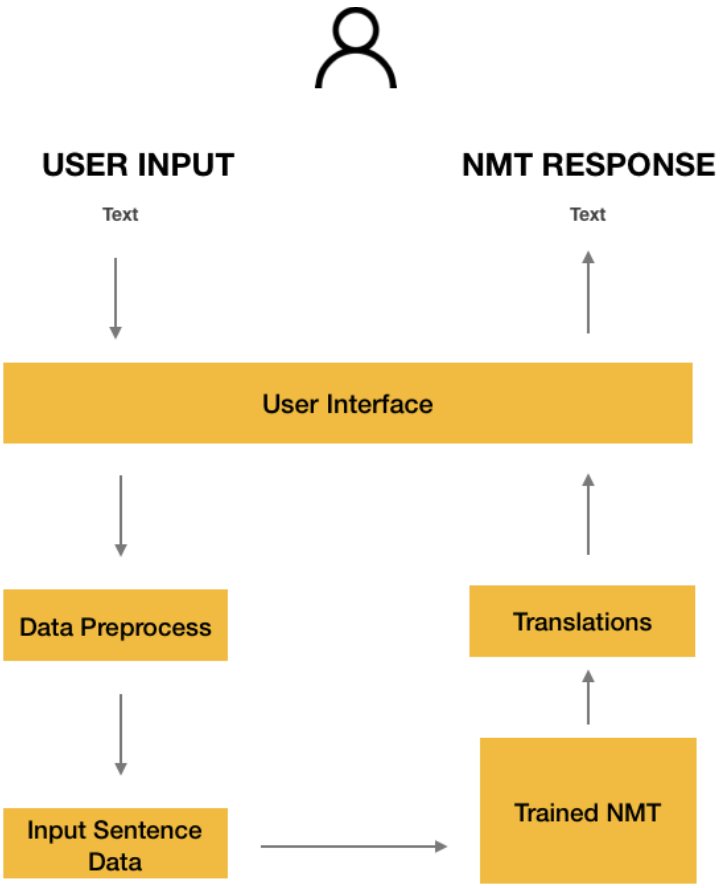



Figure 2: Workflow of the whole application

First, the user inputs the data in English. Then the data is preprocessed (tokenized and padded) and then sent into the model. The model has 1 bidirectional LSTM Encoders and 2 LSTM layers. The hidden layers size being used is 512.

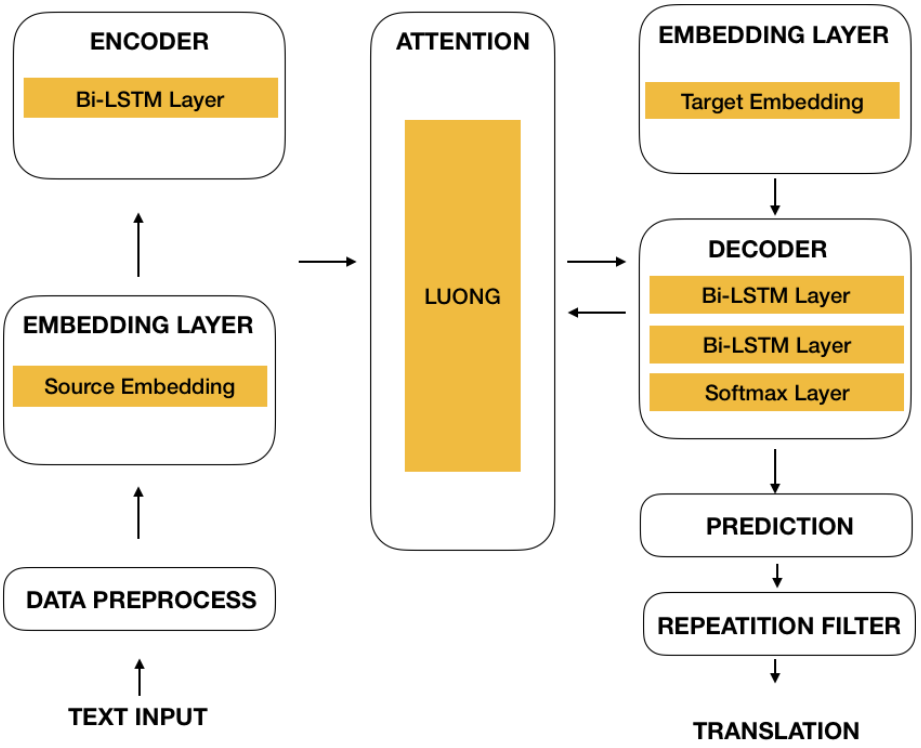



Figure 3: Deployed NMT Structure

The output from the Luong attention layer along with the target embeddings is fed into the decoder. At each time step, the translations are produced. The output of the decoder is then fed into the softmax layer. And the loss at each time step is calculated and the optimizer Adam is used to minimise the loss. In the final step translated sentences of English to French displayed back at the UI.

## Important Code Snippets:

```
# *************************
# ***** ENCODER SCOPE *****
# *************************

with tf.variable_scope("encoder"):

  if not 'bi' in self.enc_cell_name:

    enc_cell = multi_rnn_cell(self.enc_cell_name,
                              self.hid_size,
```

```
                                self.enc_layer,
                                self.keep_prob)
    enc_outputs, enc_state = tf.nn.dynamic_rnn(enc_cell,
                                               inputs=src_emb,
                                               sequence_length=src_size,
                                               dtype=tf.float32)

  else:

    stacked_biRNN_fw, stacked_biRNN_bw = multi_biRNN_pair(self.enc_cell_name,
                                                          self.hid_size,
                                                          self.enc_layer,
                                                          self.keep_prob)
    enc_outputs, enc_state = tf.nn.bidirectional_dynamic_rnn(cell_fw =
stacked_biRNN_fw,
                                                             cell_bw =
stacked_biRNN_bw,
                                                             inputs = src_emb,
                                                             sequence_length =
src_size,
                                                             dtype = tf.float32)

    # concatenate two outputs of LSTM cell as 1 tensor
    # enc_outputs = (output_fw, output_bw)
    enc_outputs = tf.concat([enc_outputs[0], enc_outputs[1]], -1)


# *************************
# ***** DECODER SCOPE *****
# *************************

with tf.variable_scope("decoder"):

  dec_cell = multi_rnn_cell(self.dec_cell_name, self.hid_size, self.dec_layer,
self.keep_prob)


  # OPTIONS FOR ATTENTION MECHANISM
  if self.attention == 'Bahdanau':

    attention_mechanism = tf.contrib.seq2seq.BahdanauAttention(
            self.hid_size,
            enc_outputs,
            memory_sequence_length = src_size)

  if self.attention == 'Luong':

    attention_mechanism = tf.contrib.seq2seq.LuongAttention(
            self.hid_size,
            enc_outputs,
            memory_sequence_length=src_size)

  # ATTENTION WRAPPER:
  # wrap dec_cell and attention mechanism
  attention_cell = tf.contrib.seq2seq.AttentionWrapper(
          dec_cell,
          attention_mechanism,
          attention_layer_size=self.hid_size)

  # use attention_cell and dynamic_rnn construct decoder
```

```
  # here rely totally on attention as the information source
  dec_outputs, _ = tf.nn.dynamic_rnn(
          attention_cell, trg_emb, trg_size, dtype=tf.float32)
```

# Analysis Of Models

| # | Size of Training Data Set | Learning -Rate | optimizer | Encoder | Decoder | Attention | # Neurons | BLEU Score | Loss |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 50,000 | 0.002 | Adam | 2 bi-LSTM | LSTM | Bahdanau | 512 | 0.33 | 0.32 |
| 2 | 100,000 | 0.002 | Adam | 1-biLSTM | 2-LSTM | Bahdanau | 256 | 0.37 | 0.34 |
| 3 | 50,000 | 0.002 | Adam | 1-biLSTM | 2-LSTM | Luong | 256 | 0.32 | 0.35 |
| 4 | 50,000 | 0.002 | Adam | 1 biLSTM | 2 LSTM | Bahdanau | 256 | 0.28 | 0.38 |
| 5 | 50,000 | 0.002 | Adam | 2 bi-GRU | GRU | Bahdanau | 512 | 0.31 | 0.42 |
| 6 | 50,000 | 0.002 | Adam | 2 bi-LSTM | 3 LSTM | Bahdanau | 256 | 0.28 | 0.43 |
| 7 | 50,000 | 0.003 | Adam | 4 bi-LSTM | 4 LSTM | Bahdanau | 512 | 0.32 | 0.45 |
| 8 | 100,000 | 0.002 | Adam | 1 bi-LSTM | 2 LSTM | Luong | 256 | 0.37 | 0.53 |
| 9 | 50,000 | 0.002 | Adam | bi-LSTM | GRU | Bahdanau | 256 | 0.35 | 0.78 |
| 10 | 50,000 | 0.002 | Adam | 2 bi-LSTM | 2 LSTM | Luong | 256 | 0.30 | 1.09 |
| 11 | 50,000 | 0.001 | Adam | 3 bi-LSTM | 2 LSTM | Luong | 256 | 0.25 | 1.20 |
| 12 | 50,000 | 0.001 | RMSprop | 4 bi-LSTM | 4 LSTM | Bahdanau | 512 | 0.41 | 1.69 |
| 13 | 50,000 | 0.003 | RMSprop | 4 bi-LSTM | 4 LSTM | Luong | 512 | 0.32 | 1.88 |
| 14 | 50,000 | 0.01 | RMSprop | 4 bi-LSTM | 4 LSTM | Luong | 512 | 0.22 | 2.24 |
| 15 | 50,000 | 0.003 | RMSprop | 2 bi-GRU | 3 LSTM | Bahdanau | 256 | 0.30 | 2.60 |
| 16 | 50,000 | 0.001 | RMSprop | 4 LSTM | 4 LSTM | Bahdanau | 512 | 0.33 | 2.82 |
| 17 | 50,000 | 0.0001 | Adam | 4 bi-GRU | 4 GRU | Bahdanau | 512 | 0.21 | 3.19 |
| 18 | 50,000 | 0.001 | RMSprop | 4 RNN | 4 RNN | Bahdanau | 512 | 0.29 | 4.22 |
| 19 | 50,000 | 0.001 | RMSprop | 4 bi-LSTM | 4 LSTM | Bahdanau | 128 | 0.41 | 4.23 |

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 20 | 50,000 | 0.002 | SGD | 3 bi-LSTM | 4 LSTM | Luong | 512 | 0.00 | 6.70 |
| 21 | 50,000 | 0.001 | RMSprop | 2 bi-LSTM | 4 LSTM | Bahdanau | 128 | 0.29 | 4.07 |
| 22 | 600,000 | 0.002 | Adam | 1-biLSTM | 2-LSTM | Luong | 256 | 0.00 | 0.70 |
| 23 | 100,000 | 0.001 | RMSprop | 4 bi-LSTM | 4 LSTM | Bahdanau | 512 | 0.38 | 1.77 |

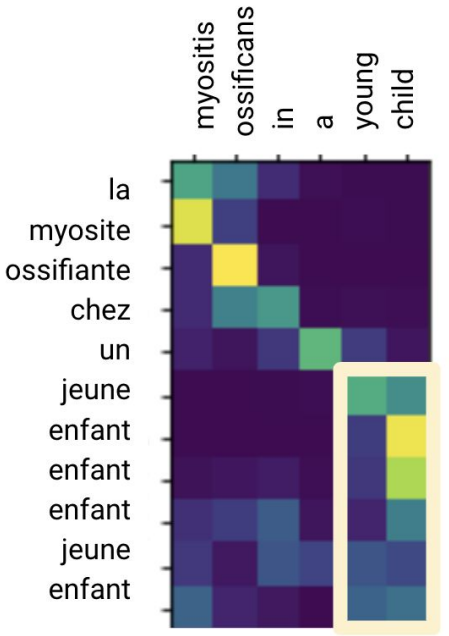



Figure 1: Attention Plot
From a shallow Network

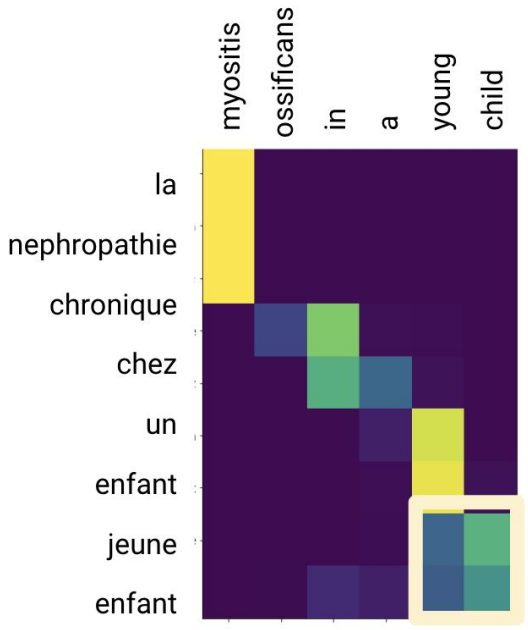



Figure 2: Attention Plot
From a bigger Network

There are many repetition of words in the translation from this network. This happen because the attention layer does not keep track of the alignment history and punish a repeated translation of the same word.

According to our observation, bigger network actually helps with this over-translation situation.

# Referenced Papers

## 1. Neural Machine Translation By Jointly Learning to Align And Translate

Link

**Breakthrough:**

For previous seq2seq NMT, a potential issue is that a neural network needs to be able to compress all the necessary information of a source sentence into a fixed-length vector.

The proposed model:

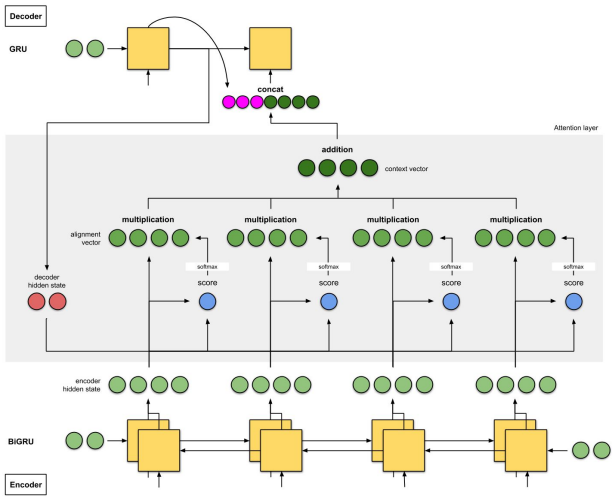
(1) extends this by allowing a model to automatically (soft-)search for parts of a source sentence that are relevant to predicting a target word, without having to form these parts as a hard segment explicitly

(2) qualitative analysis reveals that the (soft-)alignments found by the model agree well with our intuition.

**Proposed Approach:**

- the underlying framework is RNN Encoder–Decoder
- used bidirectional RNN for the encoder, In this way, the annotation contains the summaries of both the preceding words and the following words.
- introduced attention layer to calculate the context vector

### Architecture:




### Data Used:

348M words reduced from WMT ’14 (contains the following English-French parallel corpora: Europarl (61M words), news commentary (5.5M), UN (421M) and two crawled corpora of 90M and 272.5M words respectively)

## 2. Massive Exploration of Neural Machine Translation Architectures

Link

### Conclusions:

- 2048 dimensional-embeddings yielded the best result, while 128-dimensional embeddings performed well with faster convergence
- LSTMs output perform GRUs, but they are way better than RNN: This suggests that the decoder indeed passes information in its own state throughout multiple time steps instead of relying solely on the attention mechanism and current input (which includes the previous attention context).
- 2 and 4 Layers of encoders don’t show much difference for the result
- 4 layers of a decoder are better than 2-layer decoder
- Bidirectional vs. Unidirectional encoders: Bidirectional is better than uni-directional cell
- well-tuned beam search is crucial. Beam widths of 5 to 10 together with a length penalty of 1.0 seemed to work well

## 3. A Survey Of Domain Adoption Methods in Healthcare

Link

The domain-specific crop is rarely found so vanilla NMT performs poorly in

Leveraging both out of domain parallel corps as well as the monolingual crop is important.

High-quality parallel corps are only available for a few languages paired with English and several European languages.

Chinese-English domain corpus has 1M sentences but for token language domain parallel corpus is only 200k. So it's important to leverage spoken domain data with patent domain data.

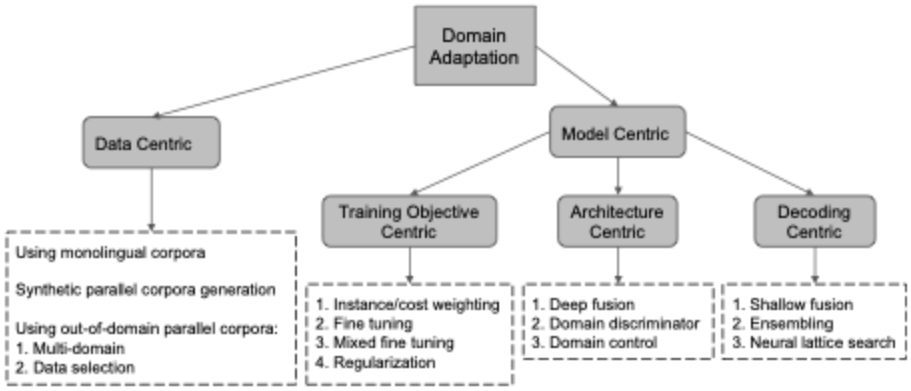



Figure 1: Overview of domain adaptation for NMT.

Fine-Tuning: It is a conventional way for domain adoption. In this, a neural machine translation system is trained on the rich out-domain corpus. Then fine tuning is applied on in domain corpora.

Mixed Fine Tuning: It is a combination of  and fine-tuning methods

Train a model until on out of domain data until convergence. Resume training the NMT model from 1 on a mix of in domain and out of domain data(by oversampling) until convergence

## 4. Fast Domain Adoption For NMT

Link

Abstract:

The basic concept is to train a large Neural Network that maximizes the translation performance on a parallel corpus. An approach for adapting the NMT system to train a new domain.

Due to recent developments, the Statistical Neural Machine learning models have become popular. There are 2 problems. First training can take weeks and training a new model combining data is difficult.Second since in domain is relatively small out domain will tend to dominate the training data.

Related Work:

The existed approaches are divided into three categories
First, the out-of-domain training data can be scored by a model built only on the in-domain training data. Based on the scores, we can either use a certain amount of best scoring out-of-domain training data to build a new translation system or assign a weight to each sentence which determines its contribution towards the training a new system

A widely used approach is to train an additional SMT system based only on the in-domain data in addition to the existing out-of-domain SMT system. By interpolating the phrase tables, the in-domain data can be integrated into the general system. In NMT, we do not have any phrase tables and can not use this method.

The third approach is called semi-supervised training, where a large in-domain monolingual data is first translated with a machine translation engine into a different language to generate parallel data. The automatic translations have been used for retraining the language model and/or the translation mode

---

## Summary of our work

All experiments are run using our own modelling framework. We mainly referenced Tensorflow Official NMT tutorial, Wu Jiaocan's tensorflow tutorials, Xue Youluo's seq2seq tutorial to build our framework based on tensorflow.

**Our Contribution:**

1. We built a flexible neural machine translation framework on tensorflow which allows easy building of NMT and allows convenient adjustment of NMT structure, cell-type, attention mechanism by simple parameter setting.
2. We have added a filter to the predicted sentence to get rid of all repeated words

In total, we have run more than 20 experiments, all details could be found in the excel summary here. Below observations and conclusions are all based on those experiments.

**Observations:**

- The number of hidden Layers
  - We observed better BLEU score achieved by network with more hidden layers
  - But as the number of hidden layer increased, the convergence takes much longer time
- Cell Type Comparison - RNN vs. LSTM vs. GRU vs Bidirectional LSTM/GRU
  - LSTM and GRU shared similar performance, and better than Basic RNN cell
  - Bidirectional Cell outputermed Unidirectional Cell
- Number of hidden neurons
  - Large number of hidden neurons converge much slower than small number of hidden neurons
  - Some network with large number of neurons could help with the over-translation issue according to our observation.
- Attention Mechanism
  - In a shallow network, Luong delivered a little better performance than Bahdanau

---

# Limitation of Current Solution and Future Improvement

## 1. Over-translation

Through translation, we discovered several word repetition problem i.e few words after translation were found more than once.
Our future goal is to build a complex system that can solve this problem. According to Modeling Coverage for Neural Machine Translation, there's a new network called coverage-based NMT to solve this over-translation problem brought about by attention-based network

## 2. Evaluation metric

BLEU, Loss, Levenshtein Distance are not perfect evaluation metrics for our modeling performance, they don't take semantics into consideration, but just compared the similarity between predicted and real translations. For example, there are multiple ways to express the same meaning.

## 3. Context Expansion

Currently our training is only limited to professional medical expressions, to apply it in a broader context, we need to add other corpus in our training scope, and leverage some transfer learning result to quickly adapt our training result into a more flexible context of usage.