Language Resources and Evaluation (2023) 57:1671–1696
https://doi.org/10.1007/s10579-023-09671-2

ORIGINAL PAPER

adaptNMT: an open-source, language-agnostic development environment for neural machine translation

Séamus Lankford[1,2] · **Haithem Afli**[2] · **Andy Way**[1]

Accepted: 26 April 2023 / Published online: 14 July 2023

Abstract

adaptNMT streamlines all processes involved in the development and deployment of RNN and Transformer neural translation models. As an open-source application, it is designed for both technical and non-technical users who work in the field of machine translation. Built upon the widely-adopted OpenNMT ecosystem, the application is particularly useful for new entrants to the field since the setup of the development environment and creation of train, validation and test splits is greatly simplified. Graphing, embedded within the application, illustrates the progress of model training, and SentencePiece is used for creating subword segmentation models. Hyperparameter customization is facilitated through an intuitive user interface, and a single-click model development approach has been implemented. Models developed by adaptNMT can be evaluated using a range of metrics, and deployed as a translation service within the application. To support eco-friendly research in the NLP space, a green report also flags the power consumption and $kgCO_2$ emissions generated during model development. The application is freely available (http://github.com/adaptNMT).

Keywords Neural machine translation · Language technology · NMT · Natural language processing · Green NLP

✉ Séamus Lankford
seamus.lankford@mtu.ie

Haithem Afli
haithem.afli@mtu.ie

Andy Way
andy.way@adaptcentre.ie

[1] ADAPT Centre, Dublin City University, Dublin, Ireland

[2] ADAPT Centre, Munster Technological University, Cork, Ireland

1 Introduction

Explainable Artificial Intelligence (XAI) (Arrieta et al., 2020; Gunning et al., 2019) seeks to ensure that the results of AI solutions are easily understood by humans. It is against this backdrop that adaptNMT has been developed to afford users a form of *Explainable Neural Machine Translation (XNMT)*. The stages involved in a typical NMT process are broken down into a series of independent steps including environment setup, dataset preparation, training of subword models, parameterizing and training of main models, evaluation and deployment. This modular approach has created an effective NMT model development process for both technical and less technical practitioners in the field. Given the environmental impact of building and running of large AI models (Henderson et al., 2020; Jooste et al., 2022b;Strubell et al., 2019), we also compute carbon emissions in a 'green report', primarily as an information aid, but hopefully as a way to encourage reusable and sustainable model development.

An important part of this research involves developing applications and models to address the challenges of language technology. It is hoped that such work will be of particular benefit to newcomers to the field of Machine Translation (MT) and in particular to those who wish to learn more about NMT.

In order to have a thorough understanding of how NMT models are trained, the individual components and the mathematical concepts underpinning both RNN- and Transformer-based models are explained and illustrated in this paper. The application is built upon OpenNMT (Klein et al., 2017) and subsequently inherits all of its features. Unlike many NMT toolkits, a CLI (command line interface) approach is not used. The interface is designed and fully implemented in Google Colab.[1] For an educational setting, and indeed for research practitioners, a Colab cloud-hosted[2] solution is often more intuitive to use. Furthermore, the training of models can be viewed and controlled using the Google Colab mobile app which is ideal for builds with long run times. GUI controls, also implemented within adaptNMT, enable the customization of all key parameters required when training NMT models.

The application can be run in local mode enabling existing infrastructure to be utilised, or in hosted mode which allows for rapid scaling of the infrastructure. A deploy function allows for the immediate deployment of trained models.

This paper is organized by initially presenting background information on NMT and related work on system-building environments in Sect. 2. This is followed by a detailed description of the adaptNMT architecture and its key features in Sect. 3. An empirical evaluation of models is carried out in Sect. 4. The system is discussed in Sect. 5 before drawing conclusions and describing future work in Sect. 6. For newcomers to the field, we suggest going straight to Sect. 3 to examine the platform's capabilities, and then discovering more about the various components and their statistical underpinning in Sect. 2. This can be followed by the remaining sections in their logical sequence.

[1] https://www.colab.research.google.com.
[2] https://www.cloud.google.com.

2 Neural networks for MT

2.1 Recurrent neural network architectures

Recurrent Neural Networks (RNNs) (Araabi & Monz, 2020; Sennrich et al., 2016a; Sennrich & Zhang, 2019) are often used for the tasks of Natural Language Processing (NLP), speech recognition and MT. RNNs, such as Long Short-Term Memory (LSTM) (Hochreiter & Schmidhuber, 1997), were designed to support sequences of input data. LSTM models use an encoder-decoder architecture which enables variable length input sequences to predict variable length output sequences. This architecture is the cornerstone of many complex sequence prediction problems such as speech recognition and MT.

RNN models enable previous outputs to be used as inputs through the use of hidden states. In the context of MT, such neural networks were ideal due to their ability to process inputs of any length. In the initial stages of NMT, the RNN encoder-decoder framework was adopted and variable-length source sentences were encoded as fixed-length vectors (Cho et al., 2014; Sutskever et al., 2014). An improvement upon the basic RNN approach was proposed in Bahdanau et al. (2014) which enhanced translation performance of the basic encoder-decoder architecture by replacing fixed-length vectors with variable-length vectors. A bidirectional RNN was now employed to read input sentences in the forward direction to produce forward hidden states while also producing backward hidden states by reading input sentences in the reverse direction. This development enabled neural networks to more accurately process long sentences, which previously had served as bottlenecks to performance, given their tendency to 'forget' words in long input sequences which are 'too far away' from the current word being processed.

More importantly, Bahdanau et al. (2014) introduced the concept of 'attention' to the basic RNN architecture, similar in spirit and intention to 'alignments' in the forerunner to NMT, Statistical MT (Och & Ney, 2003). In attention-augmented NMT, the system could now pay special heed to the most relevant other source-sentence words and use them as contextual clues when considering how best to select the most appropriate target words(s) for translationally ambiguous words in the same string.

2.2 Transformer architecture

Following the introduction of the attention mechanism, a natural line of investigation was to see whether attention could do most of the heavy lifting of translation by itself. Accordingly, Vaswani et al. (2017) proposed that "attention is all you need" in their 'Transformer architecture', which has achieved state-of-the-art (SOTA) performance on many NLP benchmarks by relying solely on an attention mechanism, removing recurrence and convolution, while allowing the use of much simpler feed-forward neural networks.

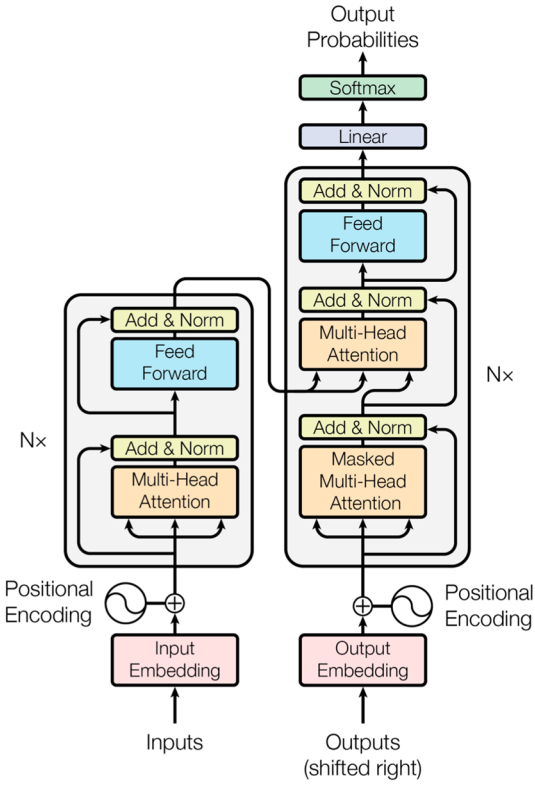


Fig. 1 The Transformer architecture using an encoder-decoder (Vaswani et al., 2017). The encoder maps an input sequence to the decoder. The decoder generates a new output by combining the encoder output with the decoder output from the previous step

This approach follows an encoder-decoder structure, and allows models to develop a long memory which is particularly useful in the area of language translation. The task of the encoder is to map an input sequence to a sequence of continuous representations, which is then passed to a decoder to generate an output sequence by using the output of the encoder together with the decoder output from the previous time step. Both the encoder and decoder each consist of a stack of 6 identical layers, whose structure is illustrated in Fig. 1. In the encoder, each layer is composed of two sub-layers: a multi-head self-attention mechanism and a fully connected feed-forward network. In the case of the decoder, there are three sub-layers: one which takes the previous output of the decoder stack, another which implements a multi-head self-attention mechanism, and the final layer which implements a fully connected feed-forward network.

2.3 Attention

As illustrated in Fig. 2, the attention function can be described as mapping a query and a set of key-value pairs to an output, where the query, keys, values, and output are all vectors. The output is computed as a weighted sum of the values, where the

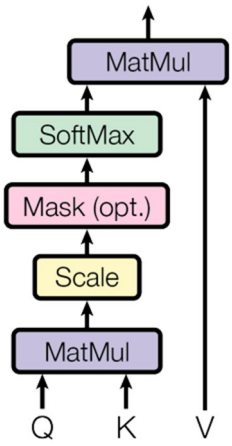


Fig. 2 Multi-Head Attention in the Decoder (Vaswani et al., 2017). In the decoder, a multi-head layer receives queries from the previous decoder sublayer, and the keys and values from the encoder output. The decoder can now attend to all words in the input sequence

weight assigned to each value is computed by a compatibility function of the query with the corresponding key, as shown in Eq. (1).

The query, keys and values used as inputs to the the attention mechanism are different projections of the same input sentence ('self-attention') and capture the relationships between the different words of the same sentence.

Both a scaled dot-product attention and a multi-head attention are used in the Transformer architecture. With scaled dot-product attention, a dot product is initially computed for each query q with all of the keys k. Subsequently, each result is divided by $\sqrt{d_k}$ and a Softmax function is applied. The process leads to the weights which are used to scale the values, v.

The Softmax function allows us to perform multiclass classification which makes it a good choice in the final layer of neural network-based classifiers. The function forces the outputs of the neural network to a total sum to 1, which can be viewed as a probability distribution across multiple classes. Therefore, Softmax is the ideal choice as the output activation function, given that NMT is essentially a multiclass classification problem where the output classes represent the words within the vocabulary.

Computations performed by scaled dot-product attention can be efficiently applied on the entire set of queries simultaneously. To achieve this, the matrices, Q, K and V, are supplied as inputs to the attention function:

$$attention(Q, K, V) = softmax(QK^T/\sqrt{d_k})V \tag{1}$$

2.4 NMT

While much research effort concentrates on creating new SOTA NMT models, excellent descriptions of the technology are also available within the literature for those starting out in the field, or for those with a less technical background (Forcada, 2017; Way, 2019).

The availability of large parallel corpora has enabled NMT to develop high-performing MT models. Breakthrough performance improvements in the area of MT have been achieved through research efforts focusing on NMT (Bahdanau et al., 2014) but the advent of the Transformer architecture has greatly improved MT performance. Consequently, SOTA performance has been attained on multiple language pairs (Bojar et al., 2017, 2018; Lankford et al., 2021b, 2022a, 2022b).

Similar to many deep-learning approaches, NMT development is underpinned by the mathematics of probability. At a fundamental level, the goal is to predict the probabilistic distribution $P(y|x)$ given a dataset D, where x represents the source input sentence and y represents the target output sentence.

Supervised training of an NMT model develops the model weights by comparing the predicted $P(y|x)$ with the correct y sentences of the training dataset, D_{Train}. In evaluating the performance of an NMT model, automatic evaluation results are determined when the predicted $P(y|x)$ sentences are compared with the correct y sentences of the test dataset, D_{Test}.

In adopting a deep learning paradigm, MT inherits the mathematical first principles which are inherent to this approach. To understand these principles, the manner in which neural networks model a conditional distribution is outlined. Furthermore, the encoder-decoder mechanism used for training NMT models is presented in the modelling subsection, and model optimization using training objectives is outlined in the learning subsection. Finally, the mathematics of how translated sentences are generated is explored in the inference subsection.

2.4.1 Modelling

In NMT, sentence-level translation is modelled using input and output sentences as sequences. Using this approach, an NMT model implements a sequence-to-sequence model with a given source sentence, $x = (x_1, \ldots, x_s)$ generating a target sentence $y = (y_1, \ldots, y_t)$.

In effect, such a sequence-to-sequence NMT model acts as a conditional language model. The decoder within the model predicts the next word of the target sentence y, while such predictions are conditioned on the source sentence x.

By applying the chain rule, a model's prediction (i.e. translation y of length T) maximizes the probability $P(y|x)$ identified in Eqs. (2) and (3):

$$P(y|x) = P(y_1|x)P(y_2|y_1, x)P(y_3|y_1, y_2, x)P(y_T|y_1, \ldots, y_{T-1}, x) \tag{2}$$

$$P(y|x) = \prod_{t=1}^{T} P(y_t|y_1, \ldots, y_{T-1}, x) \tag{3}$$

Prior to Transformer, encoder-decoder models that incorporate RNNs were the most common method of representing text sequences in NMT. RNNs are networks which accumulate information composed of similar units repeated over time. In NMT, a primary function of the RNN encoder is that it encodes text, i.e. it turns text into a numeric representation. Neurons within an RNN are illustrated in Fig. 3.

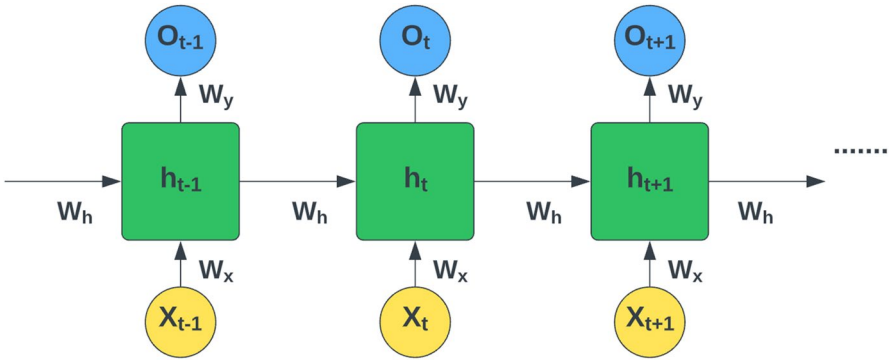


Fig. 3 Neurons within an RNN. At the input side, the neuron's input at time t is a function of the encoded word (i.e. input vector x_t) and a hidden state vector h_{t-1} which contains the previous sequence. The output generated by the neuron is represented by the vector O_t

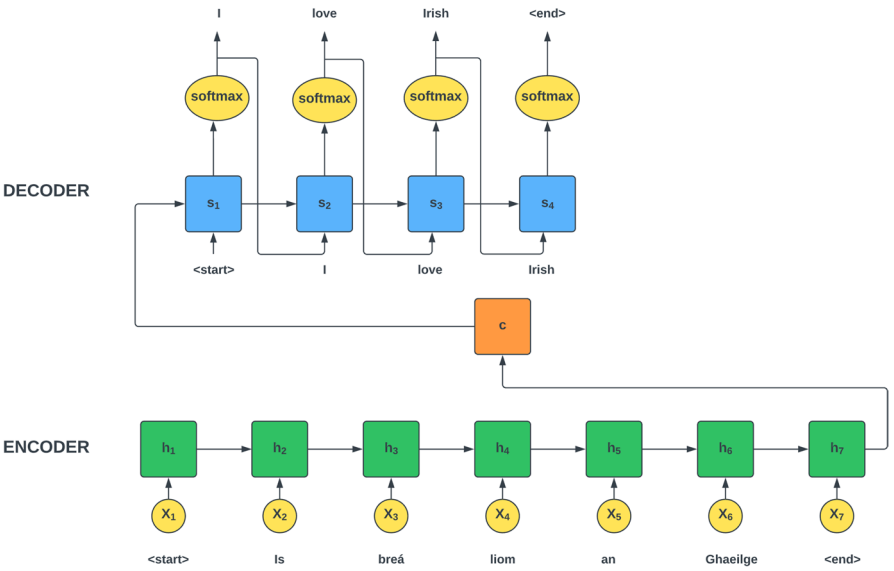


Fig. 4 Encoder-decoder architecture. The encoder encodes the entire input sequence into a fixed-length *context vector*, c, by processing input time steps. The function of the decoder is to read this *context vector* while stepping through output time steps

Decoders unfold the vector representing the sequence state and return text. An important distinction between an encoder and a decoder is illustrated in Fig. 4, where it can be seen that both the encoder hidden state and the output from the previous decoding state are required by the decoder.

To kick-start processing of the decoder, a special token <start> is used since there is no previous output. The calculations carried out by the encoder are summarized in Eq. (4):

$$h_t = RNN_{ENC}(x_t, h_{t-1}) \tag{4}$$

The RNN_{ENC} function is iteratively applied over the input sequence to generate the final encoder state, h_s which is fed to the decoder. The complete source sentence is effectively represented by h_s. The decoder within the model predicts the next word of the target sentence y, while such predictions are conditional on the source sentence x.

The RNN decoder, RNN_{DEC}, creates a state vector s_t by compressing the decoding history $y_0, \ldots, y_{t-1}$ which is described in Eq. (5). The distribution of target tokens is predicted by a classification layer which typically uses the Softmax activation function.

$$s_t = RNN_{DEC}(y_{t-1}, s_{t-1}) \tag{5}$$

2.4.2 Learning

It is possible to optimize models using different types of training objectives, although maximum log-likelihood (MLE) is the most commonly used method. Given a set of training examples $D = \{(x^s, y^s)\}_{s=1}^{S}$, the MLE is maximised according to Eqs. (6) and (7).

$$\hat{\theta}_{MLE} = \arg\max_{\theta}\{\mathcal{L}(\theta)\} \tag{6}$$

$$\mathcal{L}(\theta) = \sum_{s=1}^{S} logP(y^s|x^s);\theta) \tag{7}$$

The gradient of $\mathcal{L}$ with respect to θ is calculated using back-propagation (Rumelhart et al., 1986) as an automatic differentiation algorithm for calculating gradients of the neural network weights, where θ is the set of model parameters.

Many NMT approaches implement Stochastic Gradient Descent (SGD) as the optimization algorithm for minimising the loss of the predictive model with regard to the training data. For reasons of computational efficiency, SGD typically computes the loss function and gradients on a minibatch of the training set. The standard SGD optimizer updates parameters of an NMT model according to Equation (8), where the learning rate is specified by α:

$$\theta \leftarrow \theta - \alpha \bigtriangledown \mathcal{L}(\theta) \tag{8}$$

There are several alternatives to using SGD for optimization, among which the ADAM optimizer has proven popular due to a reduction in training times (Kingma & Ba, 2014).

2.4.3 Inference

In the context of NMT, inference should ideally find the target translated sentence y from the source x which maximizes the model prediction $P(y|x;\theta)$. However, in

Algorithm 1 Beam Search

set Beamsize = K;
$h_0 \Leftarrow encoder(S)$
$t \Leftarrow 1$
// L_S means length of source sentence;
// α is Length factor;
while $n \leq \alpha * L_S$ **do**
  $y_{1,i} \Leftarrow < EOS >$
  while $i \leq K$ **do**
    set $h_t \Leftarrow decoder(h_{t-1}, y_{t,i})$;
    set $P_{t,i} = Softmax(y_{t,i})$;
    set $y_{t+1,i} \Leftarrow argTop\_K(P_{t,i})$;
    set $i = i + 1$
  end while
  set $i = 0$
  if $h_t == < EOS >$ **then**
    break;
  end if
  set $t = t + 1$
end while
select $argmax(p(Y))$ from K candidates Y_i
return Y_i

Fig. 5 Beam search algorithm (Yang et al., 2020)

practice it is often difficult to find the translation with the highest probability due to the impractically large search space. Accordingly, to find a good but not necessarily the very 'best' (i.e. that with the highest probability given the model) translation, NMT usually relies instead on local search algorithms such as greedy search or beam search (cf. Fig. 5). Translations are carried out by default using beam search, although the option exists to switch to greedy search if needed. This approach is consistent with many other NMT tools since beam search is a classic local search algorithm. Using a pre-defined beam width parameter K, the beam search algorithm keeps only the top-K possible translations as potential candidates. With each iteration, a new potential translation is formed by combining each candidate word with a new word. New candidate translations compete with each other using log probability values to obtain the new top-K most probable results. This process is continued until the end of the translation process, and the 1-best translation is output.

2.5 Subword models

Translation by its very nature requires an open vocabulary, but restricted (e.g. 30, 50, or 70k) vocabularies are typically used for reasons of computational efficiency. However, the use of subword models aims to address this fixed vocabulary problem associated with NMT. The problem manifests itself in how previously unseen 'out-of-vocabulary' (OOV) words are handled. In such cases, a single 'UNK' (for 'unknown') token is used to 'recognize' the OOV word. Encoding rare and unknown words into sequences of subword units significantly reduces the problem and has thus given rise to a number of subword algorithms.

Optimally, this will be performed via morphological processing (Passban et al., 2018), but good quality wide-coverage morphological analysers are not always available. Therefore it is common practice to use methods such as Byte Pair Encoding (BPE) (Gage, 1994) to break down rare and previously unseen words into subword

Table 1 Key features differentiating adaptNMT from Joey NMT

Key features
AdaptNMT is built upon OpenNMT and subsequently inherits all of its features
The interface is designed and fully implemented in Google Colab
Colab is easier to follow for practitioners since each step can be executed individually. The approach is ideal in education since progression of the pipeline is demonstrated
Training of models can be viewed and controlled using Colab Android or Apple apps
adaptNMT can be run in local mode enabling existing infrastructure to be utilised or in hosted mode which allows rapid scaling of the infrastructure
Colab Pro+ provides individual researchers, or even small teams, the capacity to build large models on an excellent infrastructure with very little resources
GUI controls can split a corpus into train, validation and test datasets
GUI controls are available for hyperparameter customization in NMT training
A green report outlines the country-specific $kgCO_2$ generated when training a model
Autonotification notifies the user on completion of training
A deploy function enables the immediate deployment of trained models
The functionality of serverNMT is not available within Joey NMT

models in order to significantly improve translation performance (Kudo, 2018; Sennrich et al., 2016b).

Designed for NMT, SentencePiece (Kudo & Richardson, 2018), is a language-independent subword tokenizer that provides an open-source C++ and a Python implementation for subword units. An attractive feature of the tokenizer is that SentencePiece trains subword models directly from raw sentences.

2.6 NMT tools

Kreutzer et al. (2019) describe their Joey NMT platform[3] as a minimalist NMT toolkit, based on PyTorch, which is designed especially for newcomers to the field. Joey NMT provides many popular NMT features in a simple code base enabling novice users to easily adapt the system to their particular requirements. The toolkit supports both RNN and Transformer architectures.

Given that adaptNMT is essentially an IPython wrapper layered on top of OpenNMT, it inherits all of OpenNMT's features and continues to benefit from the work which goes into developing and maintaining its code base. adaptNMT offers a higher level of abstraction over OpenNMT where the focus is much more on usability, especially to newcomers to the field. Accordingly, it provides for easy and rapid deployment by enabling new features such as greater pre-processing, as well as GUI control over model building. It also contains green features in line with the current research drive towards smaller models with lower carbon footprints (cf. Sects. 4.4

[3] https://github.com/joeynmt/joeynmt

and 5). Such features make adaptNMT suitable for both educational and research environments. The key features differentiating adaptNMT from Joey NMT are outlined in Table 1.

Other popular frameworks for NMT system-building include FAIRSEQ[4] (Ott et al., 2019), an open-source sequence modelling toolkit based on PyTorch, that enables researchers to train models for translation, summarization and language modelling. Marian[5] (Junczys-Dowmunt et al., 2018), developed using C++, is an NMT framework based on dynamic computation graphs. OpenNMT[6] (Klein et al., 2017) is an open-source NMT framework that has been widely adopted in the research community. The toolkit covers the entire MT workflow from the preparation of data to live inference.

2.7 Hyperparameter optimization

Hyperparameters are employed in order to customize machine learning models such as translation models. It has been shown that machine learning performance may be improved through hyperparameter optimization (HPO) rather than just using default settings (Sanders & Giraud-Carrier, 2017).

The principal methods of HPO are Grid Search (Montgomery, 2019) and Random Search (Bergstra & Bengio, 2012). Grid search is an exhaustive technique which evaluates all parameter permutations. However, as the number of features grows, the amount of data permutations grows exponentially making optimization expensive in the context of developing translation models which require long build times. Accordingly, an effective, less computationally intensive alternative is to use random search which samples random configurations.

3 Architecture of adaptNMT

Having described the individual components of RNN- and Transformer-based NMT systems, we now present the adaptNMT tool itself, in which these components can be configured by the user. A high-level view of the system architecture of the platform is presented in Fig. 6. Developed as an IPython notebook, the application uses the Pytorch implementation of *OpenNMT* for training models with SentencePiece used for training subword models. By using a Jupyter notebook, the application may be easily shared with others in the MT community. Furthermore, the difficulties involved in setting up the correct development environment have largely been removed since all required packages are downloaded on-the-fly as the application runs.

There are options to run the system on local infrastructure or to run it as a Colab instance using Google Cloud. Translation models are developed using parallel text

[4] https://github.com/facebookresearch/fairseq.
[5] https://marian-nmt.github.io.
[6] https://opennmt.net.

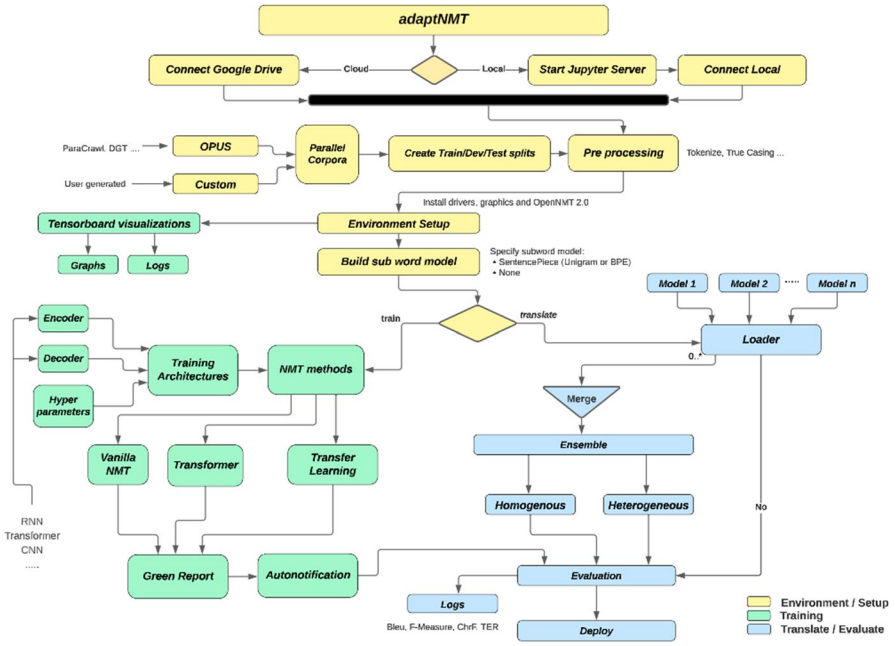


Fig. 6 Proposed architecture for adaptNMT: a language-agnostic NMT development environment. The system is designed to run either in the cloud or using local infrastructure. Models are trained using parallel corpora. Visualization and extensive logging enable real-time monitoring. Models are developed using vanilla RNN-based NMT, Transformer-based approaches or (soon) transfer learning using a fine-tuning approach. Translation and evaluation can be carried out using either single models or ensembles

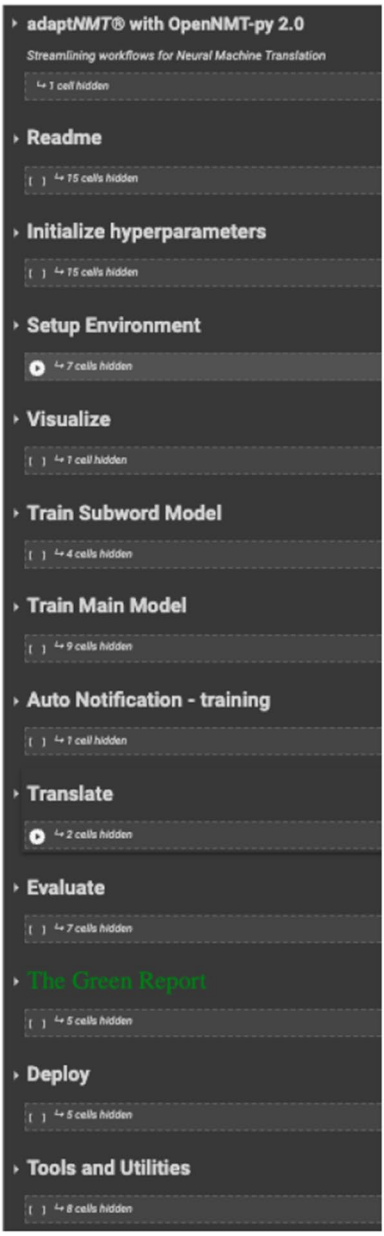
corpora of the source and target languages. A Tensorboard visualization provides a real-time graphical view of model training. The primary use-cases for the system are model building and a translation service, one or both of which can be selected at run-time. As illustrated in the system diagram in Fig. 6, generating an ensemble output while translating has also been facilitated. Models may also be deployed to a pre-configured location.

3.1 adaptNMT

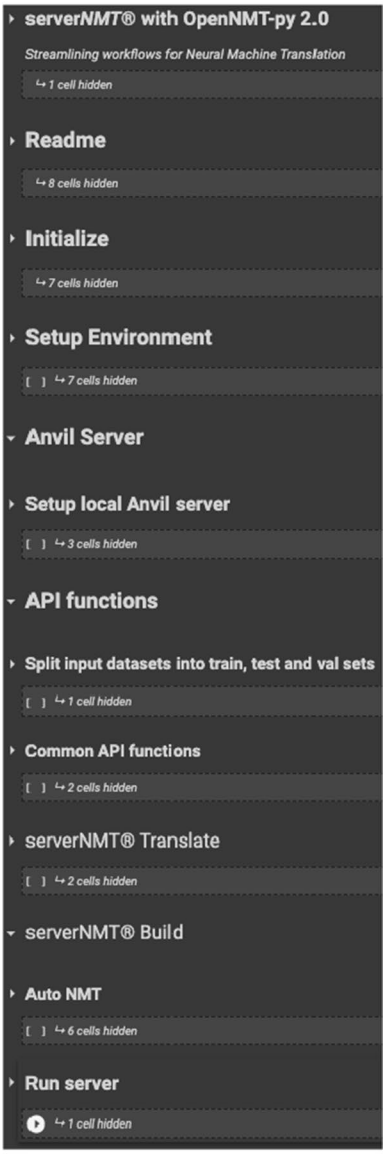
The application may be run as an IPython Jupyter notebook or as a Google Colab application. Given the ease of integrating large Google drive storage into Colab, the application has been used exclusively as a Google Colab application for our own experiments, some of which are described in Sect. 4. The key features of the notebook are illustrated in Fig. 7.

3.1.1 Initialization and logging

Initialization enables connection to Google Drive to run experiments, automatic installation of Python, OpenNMT, SentencePiece, Pytorch and other applications.



(a)



(b)

Fig. 7 adaptNMT and serverNMT. **a** Overview of adaptNMT. Key areasinclude initialization, pre-processing, environment setup, visualization, auto and custom NMT, training of subword model, training of main model, evaluation and deployment (cf. Sect. 3.1). **b** Overview of serverNMT. Highlightedcells include initialization, environmentsetup, Anvil server, API functions, translation, model building, adaptNMT and running the server (cf. Sect. 3.2)

The visualization section enables real-time graphing of model development. All log files are stored and can be viewed to inspect training convergence, the model's training and validation accuracy, changes in learning rates and cross entropy.

3.1.2 Modes of operation

There are two modes of operation: local or cloud. In local mode, the application is run so that models are built using the user's local GPU resources. The option to use cloud mode enables users to develop models using Google's GPU clusters. For shorter training times, the unpaid Colab option is adequate. However, for a small monthly subscription, the Google Colab Pro option is worthwhile since users have access to improved GPU and compute resources. Nevertheless, there are also environmental and running costs to consider (cf. Sects. 4.4 and 5), although the Google Cloud is run on a platform which uses 100% renewables (Lacoste et al., 2019). It is also a very cost-effective option for those working in the domain of low-resource languages since developing smaller models require shorter training times. However, users requiring long training times and very high compute resources will need to use their own hardware and run the application in local mode unless they have access to large budgets.

3.1.3 Customization of models

The system has been developed to allow users to select variations to the underlying model architecture. A vanilla RNN or Transformer approach may be selected to develop the NMT model. The customization mode enables users to specify the exact parameters required for the chosen approach. One of the features, AutoBuild, enables a user to build an NMT model in three simple steps: (i) upload source and target files, (ii) select RNN or Transformer, and (iii) click AutoBuild.

3.1.4 Use of subword segmentation

The type of optimizer to be used for learning can be specified, and users may also choose to employ different types of subword models when building the system. The subword model functionality allows the user to choose whether or not to use a subword model. Currently, the user specifies the vocabulary size and chooses either a SentencePiece unigram or a SentencePiece BPE subword model (cf. Sect. 2.5).

A user may upload a dataset which includes the train, validation and test splits for both source and target languages. In cases where a user has not already created the required splits for model training, single source and target files may be uploaded. The splits needed to create the train, validation and test files are then automatically generated according to the user-specified split ratio. Given that building NMT models typically demands long training times, an automatic notification

feature is incorporated that informs the user by email when model training has been completed.

3.1.5 Translation and evaluation

In addition to supporting training of models, the application also allows for translation and evaluation of model performance. Translation using pre-built models is also parameterized. Users specify the name of the model as a hyperparameter which is then subsequently used to translate and evaluate the test files. The option for creating an ensemble output is also catered for, and users simply name the models which are to be used in generating the ensemble output.

Once the system has been built, the model to be used for translating the test set may be selected. To evaluate the quality of translation, humans usually provide the best insight, but they may not always be available, do not always agree, and are expensive to recruit for experiments. Accordingly, automatic evaluation metrics are typically used, especially by developers monitoring incremental progress of systems (cf. Way (2018) for more on the pros and cons of human and automatic evaluation).

Several automatic evaluation metrics provided by SacreBleu[7]Post (2018) are used: BLEU Papineni et al. (2002), TER Snover et al. (2006) and ChrF Popović (2015). Translation quality can also be evaluated using Meteor Denkowski and Lavie (2014) and F1 score Melamed et al. (2003). Note that BLEU, ChrF, Meteor and F1 are precision-based metrics, so higher scores are better, whereas TER is an error-based metric and lower scores indicate better translation quality. Evaluation options available include standard (truecase) and lowercase BLEU scores, a sentence-level BLEU score option, ChrF1 and ChrF3.

There are three levels of logging for model development, training and experimental results. A references section outlines resources which are relevant to developing, using and understanding adaptNMT. Validation during training is currently conducted using model accuracy and perplexity (PPL).

3.2 serverNMT

A server application, serverNMT, was also developed and implemented as an IPython notebook. It can be configured to run either as a translation server or as a build server. A secure connection, implemented from serverNMT, can be made to websites hosting embedded web apps. At the core of serverNMT, there are two embedded Python web apps, one for translation services and another for developing models, both of which use the anvil.works platform.[8]

As a build server, serverNMT enables a window to the underlying cloud infrastructure in which NMT models can be trained. A web app hosted on another system may connect to this infrastructure made available by serverNMT.

[7] https://github.com/mjpost/sacrebleu.
[8] https://anvil.works.

Table 2 Hyperparameter optimization for transformer models

Hyperparameter	Values
Learning rate	0.1, 0.01, 0.001, **2**
Batch size	1024, **2048**, 4096, 8192
Attention heads	**2**, 4, **8**
Number of layers	5, **6**
Feed-forward dimension	**2048**
Embedding dimension	128, **256**, 512
Label smoothing	**0.1**, 0.3
Dropout	0.1, **0.3**
Attention dropout	**0.1**
Average Decay	0, **0.0001**

Optimal parameters are highlighted in bold (Lankford et al., 2021b)

Table 3 EN-GA train, validation and test dataset distributions

Team	System	Train (k)	Validation	Test
adapt	covid_extended	13	502	500
adapt	combined_domains	65	502	500
IIITT	en2ga-b	8	502	500
UCF	en2ga-a	8	502	500
gaHealth	en2ga	24	502	500
gaHealth	en2ga*	24	502	338

The baseline *gaHealth* system was augmented with an 8k Covid dataset provided by LoResMT2021

Table 4 GA-EN train, validation and test dataset distributions

Team	System	Train (k)	Validation	Test
IIITT	ga2en-b	8	502	250
UCF	ga2en-b	8	502	250
gaHealth	ga2en	24	502	250

The baseline *gaHealth* system was augmented with an 8k Covid dataset provided by LoResMT2021. All overlaps were removed from the *gaHealth* corpus prior to training the *gaHealth* ga2en model

Using an Anvil server embedded within serverNMT, the application continuously waits for communication to web apps and effectively enables a cloud infrastructure for NMT. Written as a REST server, it acts as an API for serving previously built models and facilities the integration of translation models with other systems.

4 Empirical evaluation

Having described the theoretical background and the tool itself, we now evaluate the effectiveness of the adaptNMT approach by training models for English-Irish (EN-GA) and Irish-English (GA-EN) translation in the health domain using the *gaHealth* (Lankford et al., 2022a) corpus.[9] All experiments involved concatenating source and target corpora to create a shared vocabulary and a shared SentencePiece subword model. To benchmark the performance of our models, the EN-GA and GA-EN test datasets from the LoResMT2021 Shared Task[10] (Ojha et al., 2021) were used. These test datasets enabled the evaluation of the *gaHealth* models since the shared task focused on an application of the health domain, namely the translation of Covid-related data. Furthermore, using an official test dataset from a shared task enables the direct comparison of our models' performance with models entered by other teams, as well as future implementations.

The hyperparameters used for developing the models are outlined in Table 2. The details of the train, validation and test sets used by our NMT models are outlined in Tables 3 and 4. In all cases, 502 lines were used from the LoResMT2021 validation dataset whereas the test dataset used 502 lines for EN-GA translation and 250 lines for GA-EN translation. Both were independent health-specific Covid test sets which were provided by LoResMT2021. There was one exception; due to a data overlap between the test and train datasets, a reduced test set was used when testing the *gaHealth* en2ga* system.

The results from the IIITT (Puranik et al., 2021) and UCF (Chen & Fazio, 2021) teams are included in Tables 5 and 6 so the performance of the *gaHealth* models can be easily compared with the findings of the participating LoResMT2021 systems. IIITT fine-tuned an Opus MT model[11] (Tiedemann & Thottingal, 2020) on the training dataset. UCF used transfer learning (Zoph et al., 2016), unigram and subword segmentation methods for EN-GA and GA-EN translation.

4.1 Infrastructure

Rapid prototype development was enabled through a Google Colab Pro subscription using NVIDIA Tesla P100 PCIe 16GB graphic cards and up to 27GB of memory when available (Bisong, 2019). All *gaHealth* MT models were trained using *adaptNMT*.

4.2 Metrics

Automated metrics were used to determine the translation quality. In order to compare against our previous work, the performance of models is measured

[9] https://github.com/seamusl/gaHealth.
[10] https://github.com/loresmt/loresmt-2021.
[11] https://github.com/Helsinki-NLP/Opus-MT.

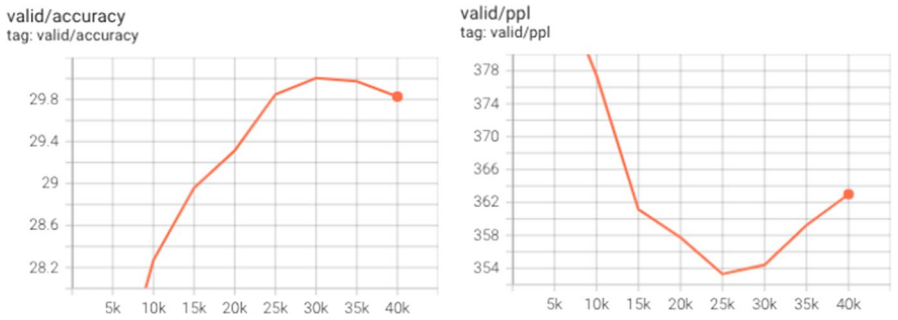


Fig. 8 adapt covid_extended system: training *EN-GA* model with 13k lines consisting of the ADAPT 5k corpus and an 8k LoResMT2021 Covid corpus. The graph on the left illustrates OpenNMT accuracy and the graph on the right demonstrates perplexity

Table 5 EN-GA *gaHealth* system compared with LoResMT 2021 EN-GA systems

Team	System	BLEU ↑	TER ↓	ChrF3 ↑
UCF	en2ga-b	13.5	0.756	0.37
IIITT	en2ga-b	25.8	0.629	0.53
adapt	combined	32.8	0.590	0.57
gaHealth	en2ga	33.3	0.604	0.56
adapt	covid_extended	36.0	0.531	0.60
gaHealth	en2ga*	**37.6**	0.577	0.57

Bold hightlights them as the winning scores

Table 6 GA-EN *gaHealth* systems compared with LoResMT 2021 GA-EN systems

Team	System	BLEU ↑	TER ↓	ChrF3 ↑
UCF	ga2en-b	21.3	0.711	0.45
IIITT	ga2en-b	34.6	0.586	0.61
gaHealth	ga2en	**57.6**	0.385	0.71

Bold hightlights them as the winning scores

using three evaluation metrics, namely BLEU, TER and ChrF. These metrics indicate the accuracy of the translations derived from our NMT systems.

Case-insensitive BLEU scores at the corpus level are reported. Model training was stopped after 40k training steps or once an early stopping criterion of no improvement in validation accuracy for four consecutive iterations was recorded.

PPL is often used to evaluate language models within NLP. It measures the effectiveness of a probability model in predicting a sample. As a metric for translation performance, it is important to keep low scores so that the number of alternative translations is reduced.

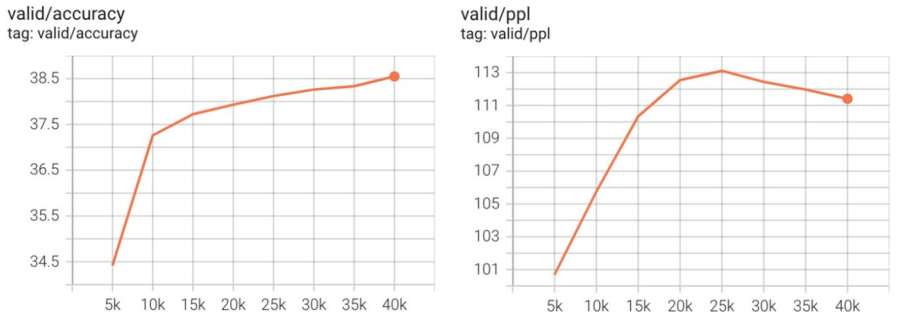


Fig. 9 *gaHealth* en2ga* system: training *EN-GA* model with combined 16k gaHealth corpus and 8k LoResMT2021 Covid corpus. The graph on the left illustrates OpenNMT accuracy and the graph on the right demonstrates perplexity

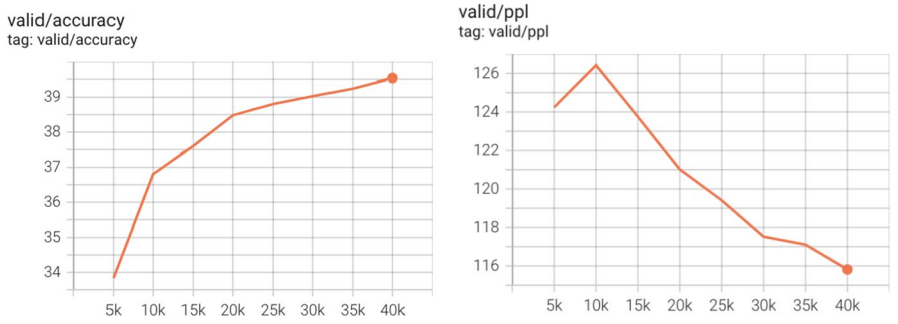


Fig. 10 *gaHealth* ga2en system: training *GA-EN* model with combined 16k gaHealth corpus and 8k LoResMT2021 Covid corpus. The graph on the left illustrates OpenNMT accuracy and the graph on the right demonstrates perplexity

4.3 Results: automatic evaluation

The experimental results from LoResMT 2021 are summarized in Tables 5 and 6. In the LoResMT2021 Shared Task, the highest-performing EN-GA system was submitted by the ADAPT team (Lankford et al., 2021a). The system uses an extended Covid dataset, which is a combination of the 2021 MT Summit Covid baseline and a custom ADAPT Covid dataset. The model, developed within adaptNMT, uses a Transformer architecture with 2 heads. It performs well across all key translation metrics (BLEU: 36.0, TER: 0.531 and ChrF3: 0.6).The training of this EN-GA model is illustrated in Fig. 8.The model achieved a maximum validation accuracy of 30.0% and perplexity of 354 after 30k steps.

The results from the LoResMT2021 Shared Task were further improved by developing models using a bespoke health dataset, *gaHealth*. Table 5 shows an improvement of 1.6 BLEU points, a relative improvement of almost 4.5%, although TER and ChrF3 scores are a little worse. Validation accuracy and PPL in training the *gaHealth* models with adaptNMT are illustrated in Figs. 9 and 10. Figure 8

Table 7 Stochastic differences between EN-GA systems

System	BLEU ↑	TER ↓	ChrF3 ↑
adaptNMT	37.6	0.577	0.570
myNMT	36.4	0.622	0.56

Table 8 Stochastic differences between EN-GA systems

System	BLEU ↑	TER ↓	ChrF3 ↑
adaptNMT	57.6	0.385	0.71
myNMT	56.6	0.399	0.703

illustrates model training using the covid_extended dataset, also developed using adaptNMT. In training the *gaHealth* en2ga* system, as highlighted in Fig. 9, the EN-GA model was trained with the combined 16k *gaHealth* and 8k LoResMT2021 corpora. The model's validation accuracy of 38.5% and perplexity of 113 achieved a BLEU score of 37.6 when evaluated with the test data.

The training of the GA-EN *gaHealth* ga2en system with the combined 16k gaHealth corpus and 8k LoResMT2021 Covid corpus is shown in Fig. 10. This model achieves a validation accuracy of 39.5% and perplexity of 116 which results in a BLEU score of 57.6. This is significantly better (by 20 BLEU points) than for the reverse direction, as it is well-known that translating into a morphologically-rich language like Irish is always more difficult compared to when the same language acts as the source. This is confirmed by comparing the results for the UCF (13.5 vs. 21.3 BLEU) and IIITT (25.8 vs. 34.6) systems in Tables 5 and 6.

Rapid convergence was observed while training the *gaHealth* models such that little accuracy improvement occurs after 30k steps, 10K fewer than for the reverse direction. Only marginal gains were achieved after this point and it actually declined in the case of the system trained using the covid_extended dataset, as the left-hand graph in Fig. 8 shows.

Of the models developed by the ADAPT team, the worst-performing model uses a larger 65k dataset. This is not surprising given that the dataset is from a generic domain of which only 20% is health related. The performance of this higher-resourced 65k line model lags behind the augmented *gaHealth* model which was developed using just 24k lines.

For translation in the GA-EN direction, the best-performing model for the LoResMT2021 Shared Task was developed by IIITT with a BLEU of 34.6, a TER of 0.586 and ChrF3 of 0.6. Accordingly, this serves as the baseline score by which our GA-EN model, developed using the *gaHealth* corpus, can be benchmarked. The performance of the *gaHealth* model offers an improvement across all metrics with a BLEU score of 57.6, a TER of 0.385 and a ChrF3 result of 0.71. In particular, the 66% relative improvement in BLEU score against the IIITT system is very significant.

4.4 Environmental impact

We were motivated by the findings of Strubell et al. (2019) and Bender et al. (2021) to track the energy consumption required to train our models. Prototype model development used Colab Pro, which as part of Google Cloud is carbon neutral (Lacoste et al., 2019). However, longer running Transformer experiments were conducted on local servers using 324 gCO_2 per kWh[12]. (SEAI, 2020). The net result was just under 10 $kgCO_2$ created for a full run of model development. Models developed during this study will be reused for ensemble experiments in the future so that work will have a life beyond this paper.

4.5 Stochastic nuances

To evaluate the translation performance of an IPython-based application such as adaptNMT, a comparison with a Python script version of the same application, myNMT.py, was conducted. We built translation models in the EN-GA and the GA-EN directions using this script. The models developed with adaptNMT were trained on Google Colab using a 12GB Tesla K80 GPU, whereas the myNMT models were trained on a local machine using a 12GB Gigabyte 3060 graphics card. The results from evaluating these models are presented in Tables 7 and 8.

Despite setting the same random seed, it is clear from Tables 7 and 8 that the translation performance of the adaptNMT models is better by 1.2 BLEU points (3.3% relative improvement) in the EN-GA direction and 1.0 BLEU point (1.8% relative improvement) in the GA-EN direction.

Given the stochastic nature of machine learning, training models on different systems can give yield different results even with the same train, validation and test data. The performance differences can be attributed to the stochastic nature of the learning algorithm and evaluation procedure. Furthermore the platforms had different underlying system architectures which is another source of stochastic error.

5 Discussion

The mathematical first principles governing NMT development were presented to demonstrate the mechanics of what happens during model training. Several parameters in Eqs. (2)–(8) are configurable within the adaptNMT application.

The environmental impact of technology, and the measurement of its effects, has gained a lot of prominence in recent years (Henderson et al., 2020). Indeed, this may be viewed as a natural response to truly massive NLP models which have been developed by large multinational corporations. In particular, HPO of NMT models can be particularly demanding if hyperparameter fine-tuning is conducted across a broad search space. As part of their work on NMT architectures, the Google Brain

[12] https://www.seai.ie/publications/Energy-in-Ireland-2020.pdf.

team required more than 250,000 GPU hours for NMT HPO (Britz et al., 2017). Training of these models was conducted using Tesla K40m and Tesla K80 GPUs with maximum power consumption between 235W and 300W, giving rise to potentially in excess of 60 MWh of energy usage. Even though the Google Cloud is carbon neutral, one must consider the opportunity cost of this energy usage.

A plethora of tools to evaluate the carbon footprint of NLP (Bannour et al., 2021) has subsequently been developed and the concept of sustainable NLP has become an important research track in its own right at many high profile conferences such as the EACL 2021 *Green and Sustainable NLP* track.[13] In light of such developments, a 'green report' was incorporated into adaptNMT whereby the $kgCO_2$ generated during model development is logged. This is very much in line with the industry trend of quantifying the impact of NLP on the environment; indeed, Jooste et al. (2022a) have demonstrated that high-performing MT systems can be built with much lower footprints, which not only reduce emissions, but also in the post-deployment phase deliver savings of almost 50% in energy costs for a real translation company.

To evaluate system performance in translating health data in the EN-GA direction, we used the adaptNMT application to develop an MT model for the LoResMT2021 Shared Task. The application was subsequently used to develop an MT model for translating in the GA-EN direction. In both cases, high-performing models achieving SOTA scores were achieved by using adaptNMT to develop Transformer models capable of generating high-quality output.

The danger of relying on increasingly large language models has been well-documented in the literature. Such discussion focuses not just on the environmental impact but also highlights the impact of in-built bias and the inherent risks that large models pose for low-resource languages (Bender et al., 2021). Using an easily-understood framework such as adaptNMT, the benefits of developing high-performing NMT models with smaller in-domain datasets should not be overlooked.

6 Conclusion and future work

We introduced adaptNMT, an application for NMT which manages the complete workflow of model development, evaluation and deployment. The performance of the application was demonstrated in the context of generating an EN-GA translation model which ranked 1st in the LoResMT2021 shared task, and validated against a standalone reimplementation of both EN-GA and GA-EN systems outside the tool, where no drop-off in performance was seen.

With regard to future work, development will focus more on tracking environmental costs and integrating new transfer learning methods. Modern zero-shot and few-shot approaches, adopted by GPT3 (Brown et al., 2020) and Facebook LASER (Artetxe & Schwenk, 2019) frameworks, will be integrated. Whereas the existing adaptNMT application focuses on customizing NMT models, a separate application

[13] https://2021.eacl.org/news/green-and-sustainable-nlp

adaptLLM will be developed to fine-tune large language models, in particular those that focus on low-resource language pairs such as NLLB (Costa-jussà et al., 2022).

The green report embedded within the application is our first implementation of a sustainable NLP feature within adaptNMT. It is planned to develop this feature further to include an improved UI and user recommendations about how to develop greener models. As an open-source project, we hope the community will add to its development by contributing new ideas and improvements.

Funding Open Access funding provided by the IReL Consortium. This research is supported by Science Foundation Ireland through ADAPT Centre (Grant No. 13/RC/2106) (www.adaptcentre.ie) at Dublin City University. This research was also funded by the Munster Technological University and the National Relay Station (NRS) of Ireland.

References

Araabi, A., Monz, C. (2020) Optimizing transformer for low-resource neural machine translation. In: Proceedings of the 28th International Conference on Computational Linguistics. International Committee on Computational Linguistics, (Online), pp. 3429–3435, https://doi.org/10.18653/v1/2020.coling-main.304.

Arrieta, A. B., Díaz-Rodríguez, N., Del Ser, J., Tabik, S., Barbado, A., Garcia, S., Gil-Lopez, S., Molina, D., Benjamins, R., Chatila, R., Herrera, F., et al. (2020). Explainable artificial intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible ai. *Information Fusion, 58*, 82–115.

Artetxe, M., & Schwenk, H. (2019). Massively multilingual sentence embeddings for zero-shot cross-lingual transfer and beyond. *Transactions of the Association for Computational Linguistics, 7*, 597–610.

Bahdanau, D., Cho. K., Bengio, Y. (2014) Neural machine translation by jointly learning to align and translate. arXiv preprint arXiv:1409.0473.

Bannour, N., Ghannay, S., Névéol, A., Ligozat, AL. (2021) Evaluating the carbon footprint of NLP methods: a survey and analysis of existing tools. In: Proceedings of the second workshop on simple and efficient natural language processing. association for computational linguistics, virtual, pp. 11–21, https://doi.org/10.18653/v1/2021.sustainlp-1.2.

Bender, EM., Gebru, T., McMillan-Major, A., Shmitchell, S.(2021) On the dangers of stochastic parrots: Can language models be too big? In: Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency. Association for Computing Machinery. FAccT '21, pp. 610–623, https://doi.org/10.1145/3442188.3445922.

Bergstra, J., & Bengio, Y. (2012). Random search for hyper-parameter optimization. *Journal of machine learning research, 13*(2), 281–305.

Bisong, E. (2019). *Building machine learning and deep learning models on google cloud patform: A Comprehensive guide for beginners*. Apress.

Bojar, O., Federmann, C., Fishel, M., Graham, Y., Haddow, B., Huck, M., Koehn, P., Monz, C. (2018) Findings of the 2018 conference on machine translation (WMT18). In: Proceedings of the Third Conference on Machine Translation: Shared Task Papers. Association for Computational Linguistics. pp. 272–303, https://doi.org/10.18653/v1/W18-6401, https://www.aclweb.org/anthology/W18-6401.

Bojar O, Chatterjee R, Federmann C, Graham, Y., Haddow, B., Huang, S., Huck, M., Koehn, P., Liu, Q., Logacheva, V., Monz, C.(2017) Findings of the 2017 conference on machine translation (WMT17). In: Proceedings of the Second Conference on Machine Translation. Association for Computational Linguistics. pp. 169–214, https://doi.org/10.18653/v1/W17-4717, https://www.aclweb.org/anthology/W17-4717.

Britz, D., Goldie, A., Luong, MT., Le, Q. (2017) Massive exploration of neural machine translation architectures. In: Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing. Association for Computational Linguistics. pp. 1442–1451, https://doi.org/10.18653/v1/D17-1151, https://aclanthology.org/D17-1151.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., & Agarwal, S. (2020). Language models are few-shot learners. *Advances in neural information processing systems, 33*, 1877–1901.

Chen, W., Fazio, B. (2021) The UCF systems for the LoResMT 2021 machine translation shared task. In: Proceedings of the 4th Workshop on Technologies for MT of Low Resource Languages (LoResMT2021). Association for Machine Translation in the Americas, Virtual, pp. 129–133, https://aclanthology.org/2021.mtsummit-loresmt.13.

Cho, K., van Merriënboer B., Bahdanau D., Bengio, Y. (2014) On the properties of neural machine translation: Encoder–decoder approaches. In: Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation. Association for Computational Linguistics. pp. 103–111, https://doi.org/10.3115/v1/W14-4012, https://aclanthology.org/W14-4012.

Costa-jussà, MR., Cross, J., Çelebi, O., Elbayad, M., Heafield, K., Heffernan, K., Kalbassi, E., Lam, J., Licht, D., Maillard, J., Sun, A., Wang, S., Wenzek, G. (2022) No language left behind: Scaling human-centered machine translation. arXiv preprint arXiv:2207.04672.

Denkowski, M., Lavie, A. (2014) Meteor universal: Language specific translation evaluation for any target language. In: Proceedings of the Ninth Workshop on Statistical Machine Translation. Association for Computational Linguistics. pp. 376–380, https://doi.org/10.3115/v1/W14-3348, https://aclanthology.org/W14-3348.

Forcada, M. L. (2017). Making sense of neural machine translation. *Translation Spaces, 6*(2), 291–309.

Gage, P. (1994). A new algorithm for data compression. *C Users Journal, 12*(2), 23–38.

Gunning, D., Stefik, M., & Choi, J. (2019). Xai–explainable artificial intelligence. *Science Robotics, 4*(37), 7120.

Henderson, P., Hu, J., Romoff, J., Brunskill, E., Jurafsky, D., & Pineau, J. (2020). Towards the systematic reporting of the energy and carbon footprints of machine learning. *Journal of Machine Learning Research, 21*(248), 1–43.

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation, 9*(8), 1735–1780.

Jooste, W., Way, A., Haque, R. (2022b) Knowledge distillation for sustainable neural machine translation. In: Proceedings of the 15th Biennial Conference of the Association for Machine Translation in the Americas (Volume 2: Users and Providers Track and Government Track). Association for Machine Translation in the Americas. pp. 221–230, https://aclanthology.org/2022.amta-upg.16.

Jooste, W., Haque, R., & Way, A. (2022). Knowledge distillation: A method for making neural machine translation more efficient. *Information*. https://doi.org/10.3390/info13020088

Junczys-Dowmunt, M., Grundkiewicz, R., Dwojak, T. (2018) Marian: Fast neural machine translation in C++. In: Proceedings of ACL 2018, System Demonstrations. Association for Computational Linguistics, Melbourne. pp. 116–121, https://doi.org/10.18653/v1/P18-4020, https://aclanthology.org/P18-4020.

Kingma, D.P., Ba, J. (2014) Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980.

Klein, G., Kim, Y., Deng, Y., Senellart, J., Rush, A. (2017) OpenNMT: Open-source toolkit for neural machine translation. In: Proceedings of ACL 2017, System Demonstrations. Association for Computational Linguistics, Vancouver. pp. 67–72, https://aclanthology.org/P17-4012.

Kreutzer, J., Bastings, J., Riezler, S. (2019) Joey NMT: A minimalist NMT toolkit for novices. In: Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th

International Joint Conference on Natural Language Processing (EMNLP-IJCNLP): System Demonstrations. Association for Computational Linguistics, Hong Kong. pp. 109–114, https://doi.org/10.18653/v1/D19-3019, https://aclanthology.org/D19-3019.

Kudo, T. (2018) Subword regularization: Improving neural network translation models with multiple subword candidates. arXiv preprint arXiv:1804.10959.

Kudo, T., Richardson, J. (2018) SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In: Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations. Association for Computational Linguistics, Brussels, Belgium, pp. 66–71, https://doi.org/10.18653/v1/D18-2012, https://aclanthology.org/D18-2012.

Lacoste, A., Luccioni, A., Schmidt, V., Dandes, T. (2019) Quantifying the carbon emissions of machine learning. arXiv preprint arXiv:1910.09700.

Lankford, S., Afli, H., Ní Loinsigh, Ó. (2022a) gaHealth: An English–Irish bilingual corpus of health data. In: Proceedings of the Thirteenth Language Resources and Evaluation Conference. European Language Resources Association, Marseille, France, pp 6753–6758, https://aclanthology.org/2022.lrec-1.727.

Lankford, S., Afli, H., Way, A. (2021a) Machine translation in the covid domain: an English-Irish case study for LoResMT 2021. In: Proceedings of the 4th Workshop on Technologies for MT of Low Resource Languages (LoResMT2021). Association for Machine Translation in the Americas, Virtual, pp. 144–150, https://aclanthology.org/2021.mtsummit-loresmt.15.

Lankford, S., Afli, H., Way, A. (2021b) Transformers for low-resource languages: Is féidir linn! In: Proceedings of Machine Translation Summit XVIII: Research Track. Association for Machine Translation in the Americas, Virtual, pp. 48–60, https://aclanthology.org/2021.mtsummit-research.5.

Lankford, S., Afli, H., & Way, A. (2022). Human evaluation of English-Irish Transformer-Based NMT. *Information, 13*(7), 309.

Melamed, I.D., Green, R., Turian, J.P. (2003) Precision and recall of machine translation. In: Companion Volume of the Proceedings of HLT-NAACL 2003 - Short Papers, Edmonton pp. 61–63, https://aclanthology.org/N03-2021.

Montgomery, D. C. (2019). *Design and analysis of experiments*. Wiley.

Och, F. J., & Ney, H. (2003). A systematic comparison of various statistical alignment models. *Computational Linguistics, 29*(1), 19–51.

Ojha, A.K., Liu, CH., Kann, K. (2021) Findings of the LoResMT 2021 shared task on COVID and sign language for low-resource languages. In: Proceedings of the 4th Workshop on Technologies for MT of Low Resource Languages (LoResMT2021). Association for Machine Translation in the Americas, Virtual, pp. 114–123, https://aclanthology.org/2021.mtsummit-loresmt.11.

Ott, M., Edunov, S., Baevski, A., Fan, A., Gross, S., Ng, N., Grangier, D., Auli, M. (2019) fairseq: A fast, extensible toolkit for sequence modeling. In: Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations). Association for Computational Linguistics, Minneapolis, Minnesota, pp. 48–53, https://doi.org/10.18653/v1/N19-4009, https://aclanthology.org/N19-4009.

Papineni, K., Roukos, S., Ward, T., Zhu, W-J. (2002) Bleu: A method for automatic evaluation of machine translation. In: Proceedings of the 40th annual meeting of the Association for Computational Linguistics, Philadelphia pp. 311–318.

Passban, P., Way, A., Liu, Q. (2018) Tailoring neural architectures for translating from morphologically rich languages. In: Proceedings of the 27th International Conference on Computational Linguistics. Association for Computational Linguistics, Santa Fe, New Mexico, USA, pp. 3134–3145, https://aclanthology.org/C18-1265

Popović, M. (2015) chrF: character n-gram F-score for automatic MT evaluation. In: Proceedings of the Tenth Workshop on Statistical Machine Translation. Association for Computational Linguistics, Lisbon, Portugal, pp 392–395, https://doi.org/10.18653/v1/W15-3049, https://aclanthology.org/W15-3049

Post, M. (2018) A call for clarity in reporting BLEU scores. In: Proceedings of the Third Conference on Machine Translation: Research Papers. Association for Computational Linguistics, Brussels, Belgium, pp. 186–191, https://doi.org/10.18653/v1/W18-6319, https://aclanthology.org/W18-6319

Puranik, K., Hande, A., Priyadharshini, R., Durairaj, T., Sampath, A., Pal Thamburaj, K., Chakravarthi, BR. (2021) Attentive fine-tuning of transformers for translation of low-resourced languages @ LoResMT 2021. In: Proceedings of the 4th Workshop on Technologies for MT of Low Resource

Languages (LoResMT2021). Association for Machine Translation in the Americas, Virtual, pp. 134–143, https://aclanthology.org/2021.mtsummit-loresmt.14.
Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature, 323*(6088), 533–536.
Sanders, S., & Giraud-Carrier, C. (2017). Informing the use of hyperparameter optimization through metalearning. *2017 IEEE International Conference on Data Mining (ICDM)* (pp. 1051–1056). IEEE.
SEAI (2020) Sustainable Energy in Ireland. https://www.seai.ie/publications/Energy-in-Ireland-2020.pdf, Retrieved March 14, 2022
Sennrich, R., Haddow, B., Birch, A. (2016a) Edinburgh neural machine translation systems for WMT 16. In: Proceedings of the First Conference on Machine Translation: Volume 2, Shared Task Papers. Association for Computational Linguistics, Berlin, Germany, pp. 371–376, https://doi.org/10.18653/v1/W16-2323, https://aclanthology.org/W16-2323.
Sennrich, R., Haddow, B., Birch, A. (2016b) Neural machine translation of rare words with subword units. In: Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). Association for Computational Linguistics, Berlin, Germany, pp. 1715–1725, https://doi.org/10.18653/v1/P16-1162, https://aclanthology.org/P16-1162.
Sennrich, R., Zhang, B. (2019) Revisiting low-resource neural machine translation: A case study. In: Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics. Association for Computational Linguistics, Florence, Italy, pp. 211–221, https://doi.org/10.18653/v1/P19-1021, https://aclanthology.org/P19-1021.
Snover, M., Dorr, B., Schwartz, R., Micciulla, L., Makhoul, J. (2006) A study of translation edit rate with targeted human annotation. In: Proceedings of the 7th Conference of the Association for Machine Translation in the Americas: Technical Papers. Association for Machine Translation in the Americas, Cambridge, Massachusetts, USA, pp. 223–231, https://aclanthology.org/2006.amta-papers.25.
Strubell, E., Ganesh, A., McCallum, A. (2019) Energy and policy considerations for deep learning in NLP. In: Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics. Association for Computational Linguistics, Florence, Italy, pp. 3645–3650, https://doi.org/10.18653/v1/P19-1355, https://aclanthology.org/P19-1355.
Sutskever, I., Vinyals, O., Le, Q.V. (2014) Sequence to sequence learning with neural networks. In: Proceedings of advances in neural information processing systems, Montréal p. 9.
Tiedemann, J., Thottingal, S. (2020) OPUS-MT – building open translation services for the world. In: Proceedings of the 22nd Annual Conference of the European Association for Machine Translation. European Association for Machine Translation, Lisboa, Portugal, pp. 479–480, https://aclanthology.org/2020.eamt-1.61.
Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., Polosukhin, I. (2017) Attention is all you need. In: Proceedings of the 31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, p. 9.
Way, A. (2018). Quality expectations of machine translation. In J. Moorkens, S. Castilho, & F. Gaspari (Eds.), *Translation quality assessment: From principles to practice* (pp. 159–178). Springer.
Way, A. (2019). Machine translation: Where are we at today? In E. Angelone, G. Massey, & M. Ehrensberger-Dow (Eds.), *The Bloomsbury Companion to Language Industry Studies* (pp. 311–332). Bloomsbury.
Yang, S., Wang, Y., Chu, X. (2020) A survey of deep learning techniques for neural machine translation. arXiv preprint arXiv:2002.07526
Zoph, B., Yuret, D., May, J., Knight, K. (2016) Transfer learning for low-resource neural machine translation. In: Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing. Association for Computational Linguistics, Austin, Texas, pp. 1568–1575, https://doi.org/10.18653/v1/D16-1163, https://aclanthology.org/D16-1163