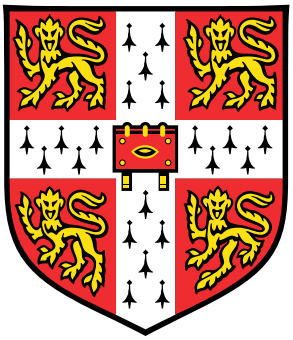
Overcoming Catastrophic Forgetting in Neural Machine Translation



Gregory Kell

Department of Engineering

University of Cambridge

This dissertation is submitted for the degree of

MPhil Machine Learning Speech and Language Technology

Wolfson College

August 2018

Declaration

I, Gregory Kell of Wolfson College, being a candidate for the MPhil in Machine Learning, Speech and Language Technology, hereby declare that this report and the work described in it are my own work, unaided except as may be specified below, and that the report does not contain material that has already been used to any substantial extent for a comparable purpose. The final word count, excluding bibliography, photographs and diagrams but including tables, footnotes, and appendices is 9906 words.

Gregory Kell
August 2018

Acknowledgements

Firstly, I would like to acknowledge my supervisor, Professor Bill Byrne, for giving me the opportunity of working on this project, as well as the invaluable guidance he has given me. He has been an excellent supervisor and has taken the time to examine my work, help me understand my work on a deeper level and provide suggestions on where I could proceed with the project. I have learned much from this process.

Secondly, I would like to thank Felix and Danielle, who have provided me with the tools and the software which I used for the project and whose guidance was crucial to the project. It should be noted that Felix developed SGNMT, while Danielle implemented EWC within tensor2tensor. I thank them sincerely for their contribution.

Abstract

Catastrophic forgetting is a problem affecting many areas in machine learning, where one model is required to perform several tasks. One area in which this phenomenon is observed is domain adaptation in neural machine translation, where a specific-domain NMT system would initially be trained on a general-domain training set and then retrained on the specific-domain training set. While this improves the performance on the specific-domain test set, the performance on the general-domain test set degrades significantly. The aim of this dissertation is to explore solutions to reduce this degradation at the decoding and training stages of the NMT system. The first solutions explored are interpolation techniques, namely fixed and Bayesian interpolation. Fixed interpolation was able to improve the performance of the NMT model on the specific-domain test set without significant degradation on the general-domain test set. These results were used to set the weights for Bayesian interpolation, which also was able to reduce degradation of performance on the general-domain test set. Finally, elastic weight consolidation, a regularisation technique, was used to retrain the NMT model on the specific-domain test set. This was also able to limit the degradation on the general-domain test set while improving the performance on the specific-domain test set after 20,000 iterations on the biology training set, although the degradation was still greater than that under any of the interpolation techniques. A further 20,000 iterations led to significant improvements on the specific-domain test set, while any reduction in degradation on the general-domain test set was lost.

Table of contents

List of figures

List of tables

Chapter 1

Introduction

1.1 Motivation

Catastrophic forgetting [14] is the phenomenon, whereby a machine learning model is trained on task A and then retrained on task B, after which it forgets much of what it originally learned on task A. This is a problem that is observed in almost all machine learning models and is a major obstacle to sequential learning, where a model learns a series of several tasks. Under this scheme, each task is followed or preceded by another task.

The aim of this work is to investigate solutions to this problem in the context of domain adaptation in neural machine translation. Domain adaptation involves leveraging out-of-domain corpora to improve in-domain translations. The reason for this is that, aside from the fact the only a few languages pairs have enough high-quality corpora for NMT, there are very few domain-specific corpora and the ones that exist are very small. Thus, the domain-specific NMT system would benefit from extra resources in the out-of-domain data. Furthermore, general-domain NMT can sometimes be improved with domain-specific data.

Domain-adaptation can be characterised as sequential learning if the domains in question are learned one after the other by the same model. Thus, the issues encountered in sequential learning, such as catastrophic forgetting, are relevant to domain-adaptation in NMT. As shown in Section 3.3, the ideal outcome can be achieved by training the specific and general-domain data together. However, this would increase the runtime and computational requirements as the model would have to be retrained from scratch on a large dataset. Whereas during sequential learning, the parameters from the general-domain model can be reused each time a model is adapted to a new domain. This only require a few additional iterations, relative to the original number of iterations used to train the model on the general-domain corpus.

An example of domain adaptation would be translating from Chinese to English in the spoken-language domain[8]. The Chinese-English patent domain parallel corpus has 1M

sentence pairs, while the spoken language domain parallel corpus has 200k sentence pairs. Thus, when performing translations for the spoken language domain, the patent language domain corpus could be used along with with the spoken language domain corpus if the patent domain models are adapted to the spoken language domain..

An NMT system can be adapted to a specific domain by training on an out-of-domain corpus and then retraining on a specific domain corpus. While this improves the performance of the model on the in-domain data, its performance on the out-of-domain data can degrade significantly. In many applications, it is desirable that the degradation of the out-of-domain corpus performance is minimal. In the example of the Chinese-English spoken-language parallel corpus, it would be desirable that the model maintains its performance on the patent parallel corpus. One real-life application would be an internet translator, which would require that the model is adapted to several domains at once.

Catastrophic forgetting is covered extensively in [19] which examines this phenomenon both in neural networks and in the context of neuroscience. The potential solutions are described within this context, as many of them are inspired by neurological processes in the brain. These approaches are divided into three categories: regularisation approaches, dynamic architectures and dual-memory systems. In addition to elastic weight consolidation, which is described below, other strategies include progressive networks [21] and modified versions of fine-tuning [10].

Progressive networks entail training a neural network on one dataset (general domain) and then adding additional "columns" to the network. The network is then trained by keeping the original activation units fixed and training the parameters on the new activation units. Each layer in the column is connected to a layer in the previous column via latteral connections.

The fine-tuning modifications are introduced in [10] which reduce the degradation in performance on the general-domain corpus. This is done using Knowledge Distillation [12]. The model that is adapted to the in-domain data through fine-tuning is the student network, while the model trained using only the general-domain data is the teacher network. Thus, the domain adaptation occurs with baseline supervision using one of two possible techniques: multi-objective fine-tuning (MCL) or multiple-output layer fine tuning (MLL).

MCL consists of training the student network on a joint objective, which is supervised by the target labels of the in-domain data, as well as the predicted labels of the teacher model on the in-domain data. It is postulated that such an objective would allow the student network parameters to converge such that the network is able to effectively predict the labels of both the general-domain and the in-domain data.

Instead of simply modifying the objective function, separate output layers can be used to translate texts from different domains, as is done in the case of MLL. The advantage

MLL has over MCL is that it can retain the performance on the general-domain data more effectively, while improving its performance on the in-domain data.

1.2 Dataset

The dataset used for this investigation came from the Scielo Corpus [16], which consists of biological and health publications. As the biological corpus is significantly smaller than the health corpus, the health corpus was chosen to be the general-domain set, while the biology corpus was the in-domain set. The reason for this is that the health domain is significantly larger than the biology domain and the biology corpus has many terms which are not encountered in the health corpus. On the other hand, there appear to be fewer terms in the health corpus which are not encountered in the biology corpus. This is explained in greater detail in Section 3.1.

The language pair used in this case is the Spanish-English language pair, where the source language was Spanish and the target language was English. There are several key differences between Spanish and English, such as the fact that English follows a subject-verb-object structure and Spanish is structure using the object-verb-subject structure. [3] This is just one example of a mismatch between the languages that come under the category of *argument structure*. Another example would be

English: Daisy went upstairs to wash **her** face.

Spanish: Daisy subió a lavarse **la** cara.

where in English, possession is indicated through the possessive determinate **her**, while in Spanish this is expressed in the reflexive verb **lavarse**.

These are just a few of the structural differences which could cause difficulties for the NMT system and could affect the quality of the reference translations, as these are done by the authors of the papers in the corpus who are not professional translators and for whom English is a foreign language.

1.3 Model

As the motivation is to explore strategies to enable sequential learning in NMT, a neural network architecture which was directly applicable to NMT had to be chosen. Although this was not essential for the purposes of this experiment, it was preferable to use a state-of-the-art

architecture to allow for comparison with other methods or strategies that are currently being tested and deployed.

While recurrent neural models, such as LSTMs [23] and GRUs [27] have been used extensively in natural language processing tasks, including machine translation, their sequential nature makes it difficult to parallelise them. The Transformer model [25] excludes recurrence and relies mainly on the attention mechanism in order to perform the translations. This leads to a model that is parallelisible. As the Transformer architecture has been shown to perform well experimentally, it was used for all the models. Furthermore, running all the experiments using the same model would allow the results of the experiments to be compared, as the only differentiating factors would be the strategies employed and the hyperparameters chosen for each strategy.

1.4 Solutions Explored

One potential solution to catastrophic forgetting is weighted interpolation, which uses an ensemble of models trained on the biology and health domains separately. Given a source sentence, the scores assigned to each hypothesis by each model in the ensemble are interpolated during search. Thus, the biology domain can benefit from information provided by the health domain, while the performance of the health domain should not degrade significantly. In order to use this solution, the domain of the test set being decoded would have to be known beforehand. In addition, the weights for each domain can be tuned using a validation set and tested using a test set. The weights can thus be selected based on the prior knowledge about the domain of the input sentence and the results of the weight-tuning experiments.

Fixed weight interpolation can be extended to make the weights adaptive [1]. This would involve calculating the posterior probability of each task given the current history and using this probability to update the state-dependent weights. These also depend on the manually-chosen state-independent weights, which are independent of the probability of the task given the current history. The calculation of the posterior probability on the tasks requires the use of Bayes' theorem, hence why it is called Bayesian interpolation. The prior probability of each task, as well as the state-independent weights, would have to be defined manually and could be based on the validation results of fixed interpolation. Thus, the fixed interpolation could be used to tune the initial weights for the Bayesian interpolation.

Instead of implementing a solution at the decoding stage, one can be implemented during training. Elastic weight consolidation (EWC) [14] involves training the model on task A and then retraining on task B, while regularising the parameters such that those parameters which

are relevant to task A are constrained more than those that are not. This would require that the loss function used when training on task B would be extended with a regularisation term which includes the optimal parameters for task A and the Fisher information [14] of those parameters. This task was originally applied to catastrophic forgetting in neural networks generally, but can be applied applied to domain adaptation in neural machine translation, where the parameters are initially trained on the general-domain corpus.

It should be noted that neither Bayesian interpolation, nor EWC have previously been applied to domain adaptation in NMT and so the results presented in this thesis are novel.

1.5 Organisation of Dissertation

Chapter 2 will outline the context in which Bayesian interpolation and elastic weight consolidation were devised. It will also summarise the results of the initial experiments, as well as their relevance to overcoming catastrophic forgetting in NMT. The Scielo task itself will be explored in chapter 3, where the dataset, the baseline NMT systems and the baseline results are discussed.

The first interpolation strategy, weighted interpolation, will be covered in chapter 4, where the mathematical formulation and results will be analysed and compared with the baseline systems.

Chapter 5 will cover Bayesian interpolation and the implementation itself will be explained, along with the modifications made to the implementation. The results will be compared with those of weighted interpolation.

The final solution to catastrophic forgetting, Bayesian interpolation, will be covered in chapter 6. This is followed by the conclusion in chapter 7 which summarises the results and suggests areas for further work.

Chapter 2

Related Work

2.1 Bayesian Interpolation

Bayesian interpolation was originally used in the context of language models (LMs) for speech recognition [1]. The original aim was to find a statically-interpolated first-pass LM that could reasonably approximate a dynamically-interpolated LM in a two-pass system. This would interpolate between each model k for each task t.

The problems associated with dynamically interpolated language models included the computation involved in accessing the larger sets of weights and combining them. This is significantly greater for dynamic interpolation than for static interpolation. The computation could be reduced by using dynamic interpolation in the second pass, while static interpolation is used in the first.

In the first pass, the statically-interpolated language model would be used to recognise words and detect unknown word sections. [15] In the second pass, the n-best candidates derived from the first pass are rerecognised and the detected sections of unknown words would be transcribed using the dynamically-interpolated model. Thus the dynamically interpolated model would be applied on fewer candidate words and would not have to detect the unknown word sections.

Although this led to a significant decrease in computation, the word error rate increased substantially. This is because dynamic interpolation [2] updates the interpolation weights on-demand using the context of each utterance, while static interpolation does not adapt the weights and in many cases does not take the context into account.

Allauzen and Riley sought to solve this problem by creating a static task-independent LM that could approximate a dynmically interpolated LM. This task-independent LM could be used for a first pass, while the task-dependent LM could be used for a second pass.

Several task-independent LMs were explored, including the uniform interpolated LM, which assumes that all the weights on each model should be uniform. This is a poor approximation, as it does not leverage the a priori task probabilities.

A slightly more sophisticated model would be the prior-weighted interpolated LM, which does take advantage of the a priori task probabilities by averaging the per-task mixture weights.

The maximum-likelihood interpolated LM chooses the task that maximises the likelihood of the task given the input sentence. The mixture weights are chosen as the maximum-likelihood task-dependent weights. While the previous two LMs are ad-hoc, this model suffers from the fact that its calculation is impractical, as a task-independent solution would require T recognition systems to be run using language models with each of the task-specific mixture weights. This is necessary so that the highest scoring solution can be determined. Furthermore, the maximum-likelihood language model does not use the a priori task information.

The maximum-likelihood LM can be extended to a maximum a posteriori LM, which does leverage the a priori task information. However, like the maximum-likelihood interpolated LM, it is impractical to calculate due to the the fact that T recognition systems would have to be run with language models which are weight using the task-specific prior probabilities multiplied by the task-dependent mixture weights.

A more practical solution would be the Bayesian interpolated LM, which averages over the task priors instead of maximising the a posteriori task-independent mixture weights. It remains practical even for very large LMs.

The experiments explored all of these LMs, except for the maximum likelihood and a posteriori LMs. as those are not practical to compute. The Bayesian interpolation method had the best performance. There was no difference between the Bayesian LM used in a two-pass strategy and the one-pass dynamically interpolated LM.

NMT models and LMs have the same form in that they are predictive left-to-right mdoels in which probabiliteis depend on the history of the hypotheses. Thus, Bayesian interpolation as derived for LMs can be applied to NMT, although there would be some difference in their interpretation. For example, in addition to being dependent on the history, the probability of each target hypothesis would be dependent on the source sentence. Despite this significant difference, Bayesian interpolation can still be applied to NMT, as will be shown.

2.2 Elastic Weight Consolidation

Elastic weight consolidation (EWC) is a regularisation strategy which can be applied during training and was originally developed by Kirkpatrick et al. [14] This was designed to overcome the problem of catastrophic forgetting, which is observed in many machine learning models, including neural networks, due to the training procedures employed. EWC allows the model to learn a parameter configuration which leads to a low error rate on both tasks A and B. This technique makes use of the parameters and Fisher information matrix after the model trained on task A during training on task B. EWC is explained in more detail in Section 6.1.

Several experiments were conducted by Kirkpatrick et al. in [14] to test the effectiveness of this technique. These are summarised below.

The first experiment was to associate random binary patterns with binary outcomes using EWC and plain gradient descent. While this example is not represent of real-word problems, it can be solved analytically, which assists the comparison of EWC and plain gradient descent. As the diagonal of the Fisher information matrix is proportional to the number of patterns observed, the learning rate decreases under EWC as more patterns are added. The aim of this experiment is to compare the memory lifetimes of the the networks trained using each techniques as more patterns are added. Kirkpatrick et al. defined a memory as retained if it exceeds a certain threshold. EWC is shown to be significantly more capable of memory retention than plain gradient descent in both the analytic and empirical cases.

The second experiment was to test EWC in a supervised learning context. A fully connected multilayer neural network was trained on several tasks in sequence. Each of the tasks involved classifying hand-written digits from the Mixed National Institute of Science and Technology. In order to create distinct tasks, the input pixels of each image were randomly shuffled. Thus, each task was of equal difficulty, but required a different solution. EWC was compared with plain stochastic gradient descent (SGD) and L2 regularisation.

SGD is an optimisation algorithm used to train neural networks where each of the weights are updated based on the gradient of the loss function. While plain gradient descent uses the entire dataset to update the weights during each iteration, SGD picks one datapoint at random from the dataset and uses that to update the weights during each iteration. [5]

L2 regularisation [17] penalises the weights of the neural network by adding $\alpha \sum_k \theta_k^2$ to the loss function, where $\alpha \geq 0$ and θ_k is representative of each weight in the neural network. As α is the same for each of the weights in the neural network, all the weights are constrained equally.

EWC was shown to retain much of the performance lost by SGD on the first task when learning additional tasks. On the other hand, EWC learned new tasks better than L2

regularisation because EWC constrains each parameter based on how important it is to the initial task. When retraining on task B, a penalisation term is added to the loss which ensures that the parameters do not move too far away from the optimal parameters on task A when training on task B. This is explained in more detail in chaper 6.

Finally, the application of EWC to sequential reinforcement learning was explored. A Deep-Q-Network was used for all the experiments and these were conducted using the Atari 2600 task set. In addition to EWC, the DQN agents were extended to infer which task was being performed. Again, EWC was compared with plain gradient descent, where it was shown that EWC retained the performance of previously learnt tasks more effectively. However, its performance did not reach that of 10 separate DQNs for each task. This could be because in this case, the Fisher information underestimates the parameter uncertainty, which leads to too much importance being placed on some parameters for the completion of a given task.

While EWC was shown to perform well in a wide variety of tasks, it has yet to be applied in the context of domain adaption for neural machine translation. Given the success of EWC in the experiments described above, it is hypothesised that EWC would be appropriate for domain adaption in NMT, which is limited by the occurrence of catastrophic forgetting. The approach taken in this work will involve training the model on the health corpus and storing the fisher information for each of the parameters after training on this task. Afterwards, the model will be retrained on the biology corpus using the parameters and fisher information matrix from the health task to regularise the parameters during training. Ideally, this should allow the model to produce high-quality translations on both tasks.

Chapter 3

Scielo Task

3.1 Baseline Tasks

3.1.1 Description of Corpus

The Scielo Corpus consists of titles and abstracts of biomedical publications, mainly from Latin America, for the following language pairs: Spanish/English, French/English and Portuguese/English. [16] The corpus is divided into two domains: biological sciences and health sciences. Throughout this work, the Spanish/English language pair was used. As the health domain is significantly larger than the biology domain, the health domain was assumed to be the general domain, while the biology domain was taken to be the specific domain.

The titles and abstracts were aligned automatically using the Geometric Mapping and Alignment tool [16], the result of which was manually validated by each of the authors of the Scielo Corpus for their native languages: Mariana Neves for Portuguese, Antonio Jimeno Yepes for Spanish and Aurelie Neveol for French.

An example of a sentence from the biology domain from the source (Spanish) and target (English) languages are:

> **English:** "Into the cranial part we founded the liver and the gallbladder while into the caudal part we founded the intestine and the pancreas."
>
> **Spanish:** "En la parte caudal se encontraba la totalidad del intestino y el páncreas, mientras que en la parte craneal se observó el hígado y la vesícula biliar."

It is noteworthy that there are specific terms that are specific to the biology domain, namely, "intestine", "pancreas" and "gallbladder".

Likewise, an example of a sentence in the health domain would be:

English: "Update in lymph node metastasis from head and neck squamous cell carcinoma: Lymph node dissection, sentinel lymph node and molecular biology techniques."

Spanish: "Actualización en metástasis ganglionar de carcinoma escamoso de cabeza y cuello: Disección ganglionar, ganglio centinela y técnicas de biología molecular."

While there are terms specific to the biological domain, such as "molecular biology", there are terms that may also be observed in the health domain, such as "head" and "neck". This suggests that any model in the biology domain would benefit from the contribution of the health domain. The biology example appears to have come from an abstract, while the health example appears to be the title of a paper.

It is noteworthy that the translations were done by the authors of the papers themselves, who are presumably not professional translators. This would explain the grammatical and stylistic errors observed in the English translations of each of the example sentences. It is very likely that there are many other errors in the English corpora. However, as these errors would affect all the experiments equally, the relative difference in the results between the experiments should remain the same. Thus, the results of these experiments would still be valid.

The example Spanish sentences contain no grammatical or stylistic errors, which is to be expected, given that the authors of the papers are native Spanish speakers.

3.1.2 Corpus Statistics

The data from the health and biology corpora are used to construct training and test sets. For future experiments, a development set can also be constructed for tuning hyperparameters.

The details of the training and test sets used for each model are shown in table 3.1. The mixed set is a concatenation of the biology and health set. As expected, the number of sentences in the same for each language in each domain. There are slightly more tokens and types for the Spanish texts than for the English ones, which could be explained by the fact that Spanish is spoken faster than English and is less dense. [20]

	Domain	Health	Biology	Biology and Health
English Training Corpus	Tokens	13518696	3093336	16612032
	Types	454274	223979	559562
	Sentences	587299	125828	713127
Spanish Training Corpus	Tokens	14967671	3381986	18349657
	Types	484950	239705	592463
	Sentences	587299	125828	713127
English Test Corpus	Tokens	108707	98998	207187
	Types	18545	20194	33331
	Sentences	5027	4028	8857
Spanish Test Corpus	Tokens	124866	109067	231744
	Types	21457	21985	37530
	Sentences	5027	4028	8857

Table 3.1 Statistics of training and tests used in NMT investigation, including the number of types, tokens and sentences.

3.2 Baseline NMT Systems

3.2.1 Neural Machine Translation

Machine translation (MT) is the process of producing target sequences in one language using the source sequence in another language. The probability of outputting a target sequence y based on the source sequence $x_1^{T_x}$ of length T_x, where the length is defined as the number of tokens in a given sequence, is

$$p(\mathbf{y}|\mathbf{x}_1^{T_x}) = \Pi_i p(y_i|y_1^{i-1}, \mathbf{x}_1^{T_x}) \text{ [9]} \tag{3.1}$$

Neural machine translation uses neural networks to perform the translations, where the architecture follows an encoder-decoder approach [7]. The encoder transforms the input sequence to a latent representation, while the decoder converts the latent representation to the output sequence. The recurrent neural network (RNN) [4] has proven popular for both the encoder and decoder, due to its ability to model seq2seq models well, although in this study the Transformer architecture is used..

In traditional statistical machine translation, a separate model would have been used to align words and phrases between the source and target languages. While NMT does not have an explicit alignment mechanism [25], the attention mechanism allow for hidden representation to be mapped to the source sentence. When decoding each target word, the attention mechanism takes a weighted mean of the context vector, depending on how relevant the given context is to the word in the given position. The result is then fed into the decoder network as the history vector. The attention mechanism is described in greater detail in section 3.2.5.

3.2.2 Metric to Assess Performance

The metric used to assess the quality of output translations produced by MT is the BLEU score [18]. While it would be ideal to have human translators to assess the quality of translators in order to assess the quality of translations, the length of time this could take would render such an approach as impractical. Thus, it is essential to have an automatic metric, which can assess the adequacy, fluency and fidelity of a translation. This requires the metric to assess how close an output translation is the reference translation produced by a human professional translator.

The authors of the BLEU score based this metric on the precision of the translation, i.e. how many words in the output translation are found in the reference translation. The problem with the raw precision is that if the output translation simply consists of a repetition of a word found in the reference translation, it could achieve a high BLEU score. For example the following example could achieve a high precision, as the word "the" is found in both reference translations [18].

Candidate: the the the the the the the.

Reference 1: The cat is on the mat.

Reference 2: There is a cat on the mat.

For this reason, the counts of each word are "clipped" to the number of words found in the reference translation, which would mean that the given candidate translation would achieve a modified unigram precision of 2/7 instead of 7/7.[18]

This modified precision can be extended to calculate the n-gram precision over an entire test corpus:

$$p_n = \frac{\sum_{C \in \{Candidates\}} \sum_{n-gram \in C} Count(n-gram)}{\sum_{C' \in \{Candidates\}} \sum_{n-gram' \in C'} Count(n-gram')} \tag{3.2}$$

First, the n-gram matches are computed for each sentence. The clipped n-gram counts for all the sentences are then summed and then divided by the number of candidate n-grams in the test corpus.

The next step was to devise a method for combining the precisions for each n-gram size. As it was found in [18] that the precision decays exponentially with respect to the size of the n-gram, the geometric mean was deemed to be suitable.

The BLEU score penalises translations which are too long through the clipped precision, which decreases with spurious words in the translation. However, the clipped precision does not penalize translations which are too short. This is done via the brevity penalty [18]:

$$BP = \begin{cases} 1 & \text{if } c > r \\ e^{1-r/c} & \text{if } c \leq r \end{cases} \tag{3.3}$$

where r is the effective reference corpus length and c is the length of the candidate translation. If the candidate translation is longer than the reference translation, the brevity penalty does not affect the final BLEU score. However, if the candidate translation is shorter than the effective reference corpus length, the translation is penalised via a decaying exponential in r/c. As the brevity penalty would have penalised shorter sentences relatively harshly, it is applied over the entire candidate and reference translations, to allow some latitude at the sentence level.

The BLEU score can now be described using

$$BLEU = BP \cdot exp\left(\sum_{n=1}^{N} w_n \log p_n\right) \tag{3.4}$$

where w_n are the weights on each of the n-gram precisions. These are typically set to $w_n = 1/N$ for each n. [18]

3.2.3 SGNMT

The Syntatically Guided Neural Machine Translation Package (SGNMT) [22] was developed at the University of Cambridge by Felix Stahlberg, Eva Hasler, Danielle Saunders and Bill Byrne. It is an NMT decoding framework, which separates the prediction and decoding operations. This strict separation makes the framework incredibly flexible, which is why it was used for decoding the NMT translations

Predictors

Predictors are scoring modules which assign scores to tokens in the target language vocabulary given the internal predictor state, the history of the source sentence and external side information. The flexibility of the predictor modules allows for scores from several diverse predictors to be combined for decoding. These scores can then be assigned to partial hypotheses when decoding each sentence. As one predictor represents a single model or constraint, each predictor has a single responsibility.

Every predictor needs to implement the following methods, where the descriptions are given ad verbatim from [22]:

- initialise(src_sentence) - the predictor state is initialised using the source sentence
- get_state() - get internal predictor state
- set_state() - set internal predictor state
- predict_next() - given the internal predictor state, produce a posterior over target tokens for the next position
- consume(tokens) - update the internal predictor state by adding the token to the current history.

Decoders

Decoders are algorithms which search for the highest scoring hypothesis, such as greedy search and beam search. The list of predictors determines how partial hypotheses are scored by implementing the methods described above. initialise() is always called before decoding a new sentence, while the search strategies are described by the remaining methods.

While the strict separation between the decoders and predictors makes the framework flexible, this comes at the expense of decoding speed. In order to compensate for this, SGNMT provides two ways of reducing the decoding time:

- The vanilla decoding strategy implements the beam search implementation in Blocks, which parallelises the decoding of all the active hypotheses using a GPU.
- Batch decoding is also implemented to process several sentences at once using beam decoding, as opposed to sequentially. Larger batches should be able to use the GPU resources more efficiently.

3.2.4 Beam Search

In order to perform the decoding, a beam search algorithm [6] was used to search the hypothesis to identify the best hypothesis. This is based on greedy decoding, which selects the hypothesis extension that maximises the log probability of the distribution over all hypothesis extensions at each time step:

$$y_i^* = argmax_{y \in V_T} p(y|y_1^{i-1}, \mathbf{x}_1^{T_x}) \tag{3.5}$$

y_i is the working hypothesis at time i, V_t is the set of all possible hypotheses, y_1^{i-1} is the sequence of previous working hypotheses and $x_1^{T_x}$ is the source sentence being translated.

This can be extended to a beam search of width N by recording N best hypotheses at each time step, as opposed to only the best.

- Declare the set of the partial hypotheses V_P to be empty
- for each value of i:
  - if i=1, evaluate $p(y|y_1^{i-1}, \mathbf{x}_1^{T_x})$ for each word in the vocabulary where y_1^{i-1} is defined as the sentence start symbol <s>
  - else, evaluate $p(y|y_1^{i-1}, \mathbf{x}_1^{T_x})$ when each word is added to each of the hypotheses in the set V_P, if the hypothesis does not end with the end symbol </s>
  - select the top N hypothesis based on $p(y|y_1^{i-1}, \mathbf{x}_1^{T_x})$ and add them to the set V_P
  - prune the other hypotheses.
  - exit when all the hypotheses end in </s>

In order to prevent numerical underflow, the probabilities are calculated logarithmically. As it stands, the model biases the algorithm to shorter hypotheses, as those would have lower probabilities. In order to counter this, the log-probabilities are normalised by the lengths of the hypotheses.

The computational time complexity of beam search is O(NL+T_x), where N is the beam width and L is the length of a given target sequence. Thus, there is a trade-off when selecting the beam width between speed and accuracy, where a greater beam width leads to more accurate, but slower translations. However, beyond a certain value, a larger beam width may lead to lower BLEU scores, but better model scores.

3.2.5 Transformer Networks

Overview

The information for this section is taken from [25], where the transformer model was originally introduced. The paper is called "Attention is all you need", as the argument was that neither convolution nor recurrence are required for sequence-to-sequence modelling.

The transformer model consists of two-basic layer types, the position-wise fully-connected layers and the multi-head attention mechanisms (figure 3.1). [25] These two layers are used to create the encoder, which accepts the embedded input and the positional encoding. As there is no recurrence or convolution in the network, the positional encoding is required to indicate the position of each input in the sentence.

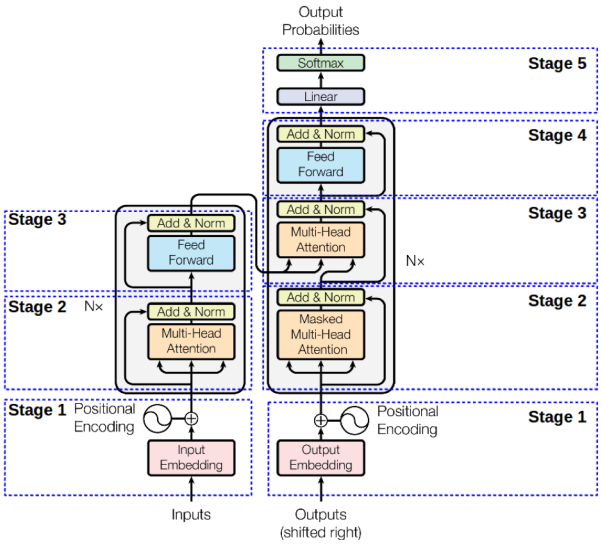


Fig. 3.1 Visualisation of the transformer architecture which was taken from [25]

The architecture of the decoder is similar, but has it an extra multi-head attention layer for the output of the encoder (stage 3). The output embedding is fed into a masked multi-head attention layer (stage 2), which ensures that the the model does not account for subsequent positions. This is because the output embedding has not yet been calculated for these positions.

Feedforward Layer

The feedforward layer consists of a linear transformation followed by a RELU function (which will be defined later and is short for Rectifying Linear Unit), which in turn is followed by another linear transformation.

$$FFN(x) = max(0, x(W_1 + b_1)W_2 + b_2) \tag{3.6}$$

A ReLU function is defined by

$$ReLU(x) = max(0, x) \tag{3.7}$$

which restricts the output to $[0, \infty]$.

The ReLU function was chosen instead of other activation functions, such as the sigmoid function, as it has been shown to be superior to those functions. [11]

Attention Layer

The attention mechanism used in the transformer network is the scaled dot-product attention:

$$Attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V \text{ [25]} \tag{3.8}$$

where the query matrix Q comes from the previous decoder layer and the keys K and values V come from the output of the encoder. This allows the decoder to attend to all the positions of the sequence outputted by the decoder. The dot-product is scaled by $\sqrt{d_k}$, which is the square root of the dimension of the keys. This prevents the dot-product from growing so large in magnitude, that the gradients of the softmax function are incredibly small.

Self-attention

The encoder contains self-attention layers, where all of the keys, values and queries are from the same sequence, namely the previous layer in the decoder. This would allow the layer to take the role of reccurrent or convolution layers by mapping from one sequence to another.

The advantage of using self-attention over recurrent layers is that there are a constant number of sequential operations, while a recurrent layer requires O(n) sequential operations [25], where n is the length of the input sequence. This means that much more of the computation can be parallelised in the case of self-attention. Furthermore, self-attention layers have a lower computational complexity than recurrent layers when the length of the

sequence is smaller than the dimensionality of the representation, which is typical when using state-of-the-art representations, such as word-piece and byte pair encoding [25].

Multi-head Attention

The basic scaled dot-product mechanism was extended to a multi-head attention mechanism, where each of the queries, keys and values were linearly transformed to h different projections. These were then fed into the attention function and the resulting heads were concatenated:

$$MultiHead(Q,K,V) = Concat(head_1,...,head_h)W^O \tag{3.9}$$

$$\text{where } head_i = Attention(QW_i^Q, KW_i^K, VW_i^V) \tag{3.10}$$

where the projections are the parameter matrices W_i^Q, W_i^K and W_i^V and W^O is the output parameter matrix for the multi-head attention layer.

Multi-head attention was chosen over single-head attention, as it "allows the model to jointly attend to information from different representation subspaces at different positions" [25]. Single-head attention does not allow for this.

Positional Encoding

The positional encoding allows the model to track the position of a given token in a sequence. This is necessary, as the model has no recurrence or convolution. The positional encodings have the same dimensionality as the embedding (d_{model}), so the two can be summed. Two examples of theses are:

$$PE_{(pos,2i)} = sin(pos/10000^{2i/d_{model}}) \tag{3.11}$$

$$PE_{(pos,2i+1)} = cos(pos/10000^{2i/d_{model}}) \tag{3.12}$$

where *pos* is position and *i* is dimension. This encoding would allow for the relative positions to be accounted for in the model as PE_{pos+k} would almost be correlated linearly with PE_{pos} because the frequency of the *sin* and *cos* functions is the low for some i.

3.2.6 Configuration

The models used in this investigation all made use of the base transformer model [25] provided in Tensor2tensor [24], with a batch size of 4096. The SGNMT framework [22]

was used to decode the hypotheses, where the beam width was set to 4 and an average of 20 checkpoints was used. All the models were trained for 125,000 iterations, apart from the health-then-biology model which was trained on the health set for 125,000 iterations and then the biology for 125,000 iterations. This is not strictly necessary, as this model can achieve a similar performance on the biology test set with fewer iterations on the biology set.

3.3 Baseline Results Across Domains and Over Combined Data

The BLEU scores of the baseline NMT systems (figure 3.2) show that, as expected, the systems trained using only the health or biology corpora performed best on their respective test set. The ideal scenario would be to train a system on a corpus consisting of both health and biology text, but this would be computationally expensive if the system were adapted to another domain. With sequential learning, the parameters of the model trained on the health domain are already optimised after a large number of iterations. The model would require significantly iterations when adapting to the biology domain.

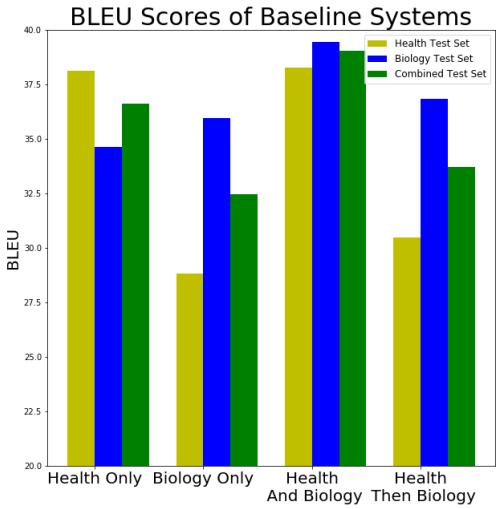


Fig. 3.2 Comparison of baseline models evaluated on the biology and health test sets. The models used include those trained using only the health and biology training sets, as well as that which was trained on a mixture of the health and biology sets. The model trained on the health set and then the biology set shows how catastrophic forgetting leads to a decrease in the BLEU score on the health test set.

When the NMT system is first trained on the health corpus and then the biology corpus, the BLEU score of the biology test set increases from the system trained only on the biology corpus. On the other hand, the BLEU score of the health test set is worse than the system trained only on the health corpus. While the performance on the biology set benefits from the information provided by the health set, the NMT system "forgets" what it originally learned from the health corpus after retraining.

The aim will be to improve the performance of the NMT system on the biology test set, without sacrificing the gains made on the health test set. Ideally, the final system will perform as well as the mixed training set, where the score on the biology test set increased significantly from the model trained using only the biology corpus. The performance on the health set, however, did not change at all compared with the model trained only on the health set. Thus, the final system should be expected to maintain the performance on the health test set.

It is noteworthy that the model trained using only the health-only corpus performs significantly better on the biology test set than that trained on the biology-only corpus when decoding the health set. This justifies the choice of the health domain as the general domain and the biology domain as the specific domain.

Chapter 4

Weighted Interpolation

4.1 Mathematical Formulation

Weighted interpolation entails decoding each source sentence by interpolating the probabilities assigned to each hypothesis by each model in the ensemble. Decoding with fixed-weight interpolation for a task t using models k, the translation objective translation of source sentence $\mathbf{x}_1^{T_x}$ is

$$y_{i,t}^* = argmax_{y \in V_T} \sum_k \lambda_{k,t} p_k(y|y_1^{i-1}, \mathbf{x}_1^{T_x}) \tag{4.1}$$

where $\lambda_{k,t}$ is the weight assigned to model k for task t and the maximum hypothesis y is chosen from a set of all possible hypotheses V_T. In other words, the weight on each model was fixed manually, depending on whether the task was the translation of the health test set or the biology test set. When fixing the weights, they should be tuned using a validation data set and then tested using a separate test set. This basic approach was originally developed for Bayesian interpolation [1] and the results are used to tune the weights $\lambda_{k,t}$ for the Bayesian interpolation in chapter 4.

4.2 Scielo Results Under Interpolation

The results of applying weighted interpolation are shown in figure 4.1. The health-only and biology-only models, where one model in the ensemble has a weight of 1 and the other has a weight of 0, are included for reference. Weighted interpolation appears to improve the BLEU score on the biology test set from the biology-only model. However, while equally weighting the biology and health set leads to a significantly higher BLEU score on the biology test set,

the score decreases as the weights are changed to favour the health-only model. Furthermore, the score on the health test set is highest when only the health-only model is used. This confirms the results of the baseline experiments, where there was no change between the BLEU score of the health-only and mixed models when decoding the health test set.

Adjusting the weights on the health set between the ranges of 0.5 and 1.0 leads to a trade-off between the performance on the biology set and the health set. Thus, if the domain is unknown it would be recommendable to use a health model with weights this range, such as in table 4.1, where the same model can be applied to both domains.

	Models k		
Tasks t		Health	Biology
	Health	0.7	0.3
	Biology	0.7	0.3

Table 4.1 Value of λ which was deemed to be a suitable trade-off off in the performance between the health and biology domains.

However, if the domain of the model is known, a text in the biology domain should be decoded with an ensemble, where each model is equally weighted, while the health domain should be decoded using only the health-only model. These results are used in the subsequent Bayesian interpolation experiments.

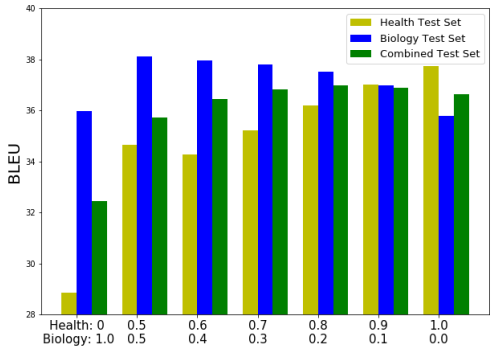


Fig. 4.1 Comparison of fixed weight interpolation using several weights on each model. The model where the health system has a weight of 0.0 and biology has a weight of 1.0 was only trained on the biology corpus.

Chapter 5

Bayesian Interpolation

5.1 Mathematical Formulation

Bayesian interpolation predicts which task is being executed and uses this information, along with the state-independent mixture weights $\lambda_{k,t}$ to calculate the state-dependent mixture weights on each model k.

Initially, the probability for the hypothesis, given the history y_1^{i-1}, the input $\mathbf{x}_1^{T_x}$ and the task t is considered (equation 5.1).

$$p(y_i|y_1^{i-1},t,\mathbf{x}_1^{T_x}) = \sum_k \lambda_{k,t} p_k(y|y_1^{i-1},\mathbf{x}_1^{T_x}) \tag{5.1}$$

However, instead of manually determining the task, the model predicts the task given the history by computing $p(t|x_1^{T_x})$ using the prior probabilities of the tasks $p(t)$ and the probabilities of each token given a history $p(y_j|y_1^{j-1},t)$. Thus, these equations can be combined to derive the state-dependent mixture weights $\alpha_{k,y_1^{j-1}}$, which are a weighted average of the state-independent mixture weights $\lambda_{k,t}$ over the distribution of the tasks given the history $p(t|y_1^{j-1})$, as shown in equation 16.

$$
\begin{aligned}
p(\mathbf{y}|\mathbf{x}_1^{T_x}) &= \Pi_i \sum_{t=1}^{T} p(t|y_1^{i-1})\, p(y_i|y_1^{i-1}, t, \mathbf{x}_1^{T_x}) && (5.2)\\
&= \Pi_i \sum_{t=1}^{T} p(t|y_1^{i-1}) \left[\sum_k \lambda_{k,t} p_k(y_i|y_1^{i-1}, \mathbf{x}_1^{T_x}) \right] && (5.3)\\
&= \Pi_i \sum_k \alpha_{k,y_1^{i-1}}\, p_k(y_i|y_1^{i-1}, \mathbf{x}_1^{T_x}) && (5.4)\\
\alpha_{k,y_1^{i-1}} &= \sum_{i=1}^{T} p(t|y_1^{i-1}) \lambda_{k,t} && (5.5)
\end{aligned}
$$

The a posteriori probabilities of the tasks can be computed using Bayes' theorem

$$p(t|y_1^{i-1}) = \frac{p(y_1^{i-1}|t)p(t)}{\sum_{t=1}^{T} p(y_1^{i-1}|t)p(t)} \tag{5.6}$$

and the probability of the history given the task $p(y_1^{i-1}|t)$ can be computed using the probability of a token, given the task and the history.

$$p(y_1^{i-1}|t) = \Pi_j^i \sum_k \lambda_{k,t} p_k(y_j|y_1^{j-1}, \mathbf{x}_1^{T_x}, t) \tag{5.7}$$

These equations allow the estimate task to vary based on the history y_1^{j-1}, which is used to update the state-dependet mixture weights $\alpha_{k,y_1^{j-1}}$. Thus, the only hyperparameters that need to be defined are the initial state-dependent mixture weights α_{k,y_1^1} and the state-independent mixture weights $\lambda_{k,t}$.

5.2 Implementation

In this section the original SGNMT implementation will be described, followed by the modifications made to the model. SGNMT is implemented in the python programming language.

5.2.1 Default

Bayesian interpolation is implemented as a function in SGNMT, where it can either reevaluate all the timesteps at once or incrementally. The SGNMT can evaluate the state-dependent weights after all of the token probabilities $p_k(y_j|y_1^{j-1}, \mathbf{x}_1^{T_X}, t)$ have been calculated, or as each of the token probabilities for the target sequence are being calculated.

Under this implementation, the state-independent mixture weights are defined as an identity matrix

$$\lambda_{k,t} = \begin{cases} 1, & \text{if } k = t \\ 0, & \text{otherwise} \end{cases}$$

Thus, each mixture weight would only be influenced by the token probabilities from one model and not the other. The consequence of this could be that the weight on model corresponding to a given task would converge to 1, while the weight on the other model would converge to 0. As is shown from the fixed-weight results for the health set, this would be the ideal configuration for this setting.

Furthermore, $p_k(y_j|y_1^{j-1}, \mathbf{x}_1^{T_X}, t)$ is stored in the variable p and the manually defined predictor weight is stored in the variable w. Both of these variables are scalars. The mathematical operations are conducted using the numpy package which is imported as np. Hence, the logarithms are implemented using np.log().

The following code appends the priors $p(t)$ to a list-variable **alphas**, which is to store the context-dependent mixture weights $\alpha_{k,y_1^{i-1}}$.

```
for (p, w) in score_breakdown[0]:
    alphas.append(np.log(w))
```

All of the $\log p_k(y_j|y_1^{j-1}, \mathbf{x}_1^{T_X}, t)$ are then appended to **alphas**, which would be equivalent to adding $p(y_1^{i-1}|t) = \Pi_j^i p(w_j|y_1^{j-1}, t)$ to $p(t)$. As the state-dependent mixture weights matrix was chosen to be the identity matrix. the token probabilities were only appended to the weights corresponding to the models from which they came.

```
for pos in score_breakdown:
    for k, (p, w) in enumerate(pos):
        alphas[k] += p
```

This results with the unnormalised posterior of t given y_1^{i-1}. Which is why $\sum_{t=1}^{T} p(y_1^{i-1}|t)p(t)$ is computed using

```
alpha_part = utils.log_sum(alphas)
```

and is used to calculate the weighted score of each hypothesis for a given position in a sequence $\sum_k \alpha_{k,y_1^{i-1}} p_k(y_i|y_1^{i-1}, \mathbf{x}_1^{T_x})$.

```
scores = [alphas[k] - alpha_part + p for k, (p, w) in enumerate(pos)]
```

5.2.2 Modification

In order to change the state-dependent mixture weights to 0.5 on each model, the following modification was made

```
for pos in score_breakdown:
    for k, (p, w) in enumerate(pos):
        for i in range(len(alphas)):
            alphas[i] += p  + np.log(lambdas[i])
```

where a new variable **lambdas** was defined to represent $\lambda_{k,t}$. Thus, when task t is the decoding of the biology test set, the **lambdas** are initialised in the following manner

```
lambdas = [0.5, 0.5]
```

This would create allow each of models to be equally weighted which was found to be the optimal weighting for the biology domain in chapter 4.2.

As this created a nested for loop, this added an extra order of magnitude to computational complexity. This follows equation (13), where the probability of each token $p_k(y_i|y_1^{i-1}, \mathbf{x}_1^{T_x})$ was multiplied by the corresponding state-independent weight $\lambda_{k,t}$.

Thus, instead of being O(n), it is now O(n^2), where n is the length of the sentence. In future work, it may be desirable to explore any changes that could be made to make this operation more efficient.

5.3 Results Under Bayesian Interpolation

In order to compare the performance of fixed weight interpolation to that of Bayesian interpolation the initial weights on the models trained on the health and biology models were chosen to be 0.5/0.5, 0.6/0.4 and 0.7/0.3, respectively. This was deemed to provide a sufficient range of initial mixture weights to examine the performance of Bayesian interpolation. The results are shown in figure 5.1.

When the models are equally weight, the biology test set produces the following example

> the family of Gramines (Poaceae) is well known from the taxonomic point of view , with more than 10,000 described species , characterized and classified according to a variety of morphological characters , particularly the inflorescence and spikelet , as well as a great number of micromorphological and molecular characters .

When the health model has a prior of 0.7 and the biology model has a prior of 0.3, the biology test set produces the following sentence

> the Gramineae (Poaceae) family is well known from the taxonomic point of view , with more than 10,000 described species , characterized and classified according to a variety of morphological characters , particularly the inflorescence and spikelet , as well as the number of micromorphological and molecular characters .

The difference between the two sentences is that the first sentence contains '... as well as a great number of micromorphological ...', while the second contains '... as well as the number of micromorphological ...'. While this is a seemingly subtle difference, it has a significant effect on the meaning of the sentence, as the former phrase suggests that the micromorphical and molecular characters themselves have been characterized and classified. The latter suggests that the **number** has been characterized and classified. The first sentence is closer than the second to the original sentence in Spanish

> la familia de las Gramíneas (Poaceae) es bien conocida desde el punto de vista taxonómico , con más de 10.000 especies descriptas , caracterizadas y clasificadas de acuerdo con una variedad de caracteres morfológicos , en particular de la inflorescencia y de la espiguilla , como también por una gran cantidad de caracteres micromorfológicos y moleculares .

as the phrase in question is '... una **gran** cantidad de caracteres micromorfológicos ...' which translates as '... a **great** quantity of morphological characters ...'.

This sentence is an example of how the performance of the biology decoding degrades as the prior weight on the biology model is reduced and that on the health model is increased.

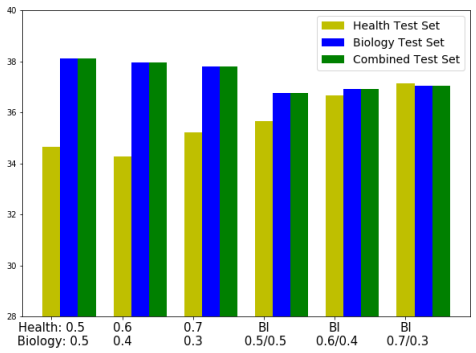


Fig. 5.1 Comparison of fixed weight interpolation with Bayesian interpolation. The initial weights used for Bayesian interpolation are 0.5/0.5, 0.6/0.4 and 0.7/0.3.

Figure 5.1 shows that Bayesian interpolation performs comparably to fixed weight interpolation when the domains of the test set are known, albeit with a greater bias to the biology set than the health set. The reason for this could be, as per the analysis in section 5.1 that the $\alpha_{k,y_1^{i-1}}$s are converging to 1 for one model and 0 for another, depending on the task. As was discovered previously, this would improve the performance on the health set, while the biology domain would benefit from $\alpha_{k,y_1^{i-1}}$ being 0.5 on both models.

Figure 5.2, which shows a plot of $\alpha_{k,y_1^{i-1}}$ for each model, confirms this. As the model in this case was used to decode the biology test set, $\alpha_{k,y_1^{i-1}}$s converges to 1.0 for the biology model and 0.0 for the health. Thus, the decoding of the final few words are influenced only by the biology model.

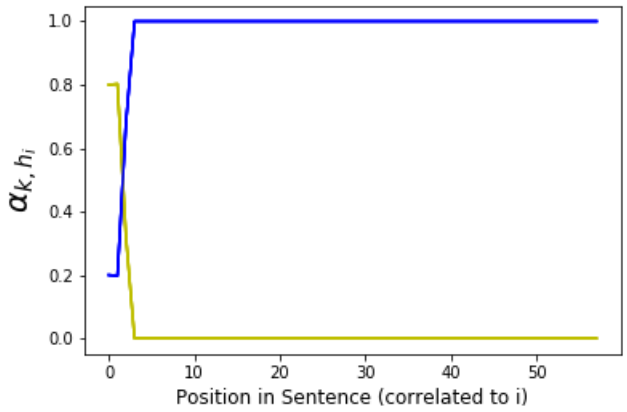


Fig. 5.2 Bayesian Interpolation weights $\alpha_{k,k,y_1^{i-1}}$ in generating the translation 'the grass family (Poaceae) is unusually well known taxonomically , with over 10,000 described species . ....'. y_1^{i-1} has been replaced with h_i for legibility.

While this may be ideal for the health set, the biology set would benefit from the contribution from the biology set. Thus, $\lambda_{k,t}$ was then set to be 0.5 for both models when translating the biology corpus. This led to a BLEU score of 38.63, which was the highest so far for the biology set. Hence, the optimal value of λ would be

	Models k		
Tasks t		Health	Biology
	Health	1.0	0.0
	Biology	0.5	0.5

Table 5.1 Optimised λ, where the rows are the tasks and the columns are the models.

In order to implement this scheme, the prior probabilities on the task would have to be defined. Instead of defining them manually, a language model can be trained for each domain and these LMs can be interpolated [13]. The interpolation weights would be adjusted to maximise the score on each domain and would be stored for each domain. The outputs from each interpolated LM for each sentence can be used to estimate the prior on each model. Furthermore the entropy from the max-entropy model [13] could be used to estimate the prior probabilities on the task. In addition, the score generated for each class in the case of the perceptron classifier can be used to estimate these probabilities [26]. All of these remain areas for further work.

Chapter 6

Elastic Weight Consolidation

6.1 Mathematical Formulation

EWC regualrises the parameters relevant to the previously learnt task by adding a penalty term to the loss function [14].

The constraint is chosen based on the Bayesian analysis of neural networks, where the posterior over the parameters $\log p(\theta|D)$ is given by

$$\log p(\theta|D) = \log p(D|\theta) + \log p(\theta) - \log p(D) \tag{6.1}$$

In this equation, the loss function L is equal to $-\log p(D|\theta)$. Thus, the constraint can be taken to be the prior term $\log p(\theta)$, while the probability of the data $\log p(D)$ can be assumed to be redundant.

It is assumed that the data can be split into two independent task, where D_A defines task A (general domain data) and D_B defines task B (in-domain data). The posterior over the parameters can then be interpreted as

$$\log p(\theta|D) = \log p(D_B|\theta) + \log p(\theta|D_A) - \log p(D_B) \tag{6.2}$$

Under this analysis, $\log p(\theta|D)$ accounts for the entire dataset, while only depending on $\log p(D_B|\theta)$, as the model has already been trained on task A. This yields the posterior of the model given task A $\log p(\theta|D_A)$ which can be assumed to be the a priori probabilities of the parameters.

As the calculation of the posterior is intractable, [14] approximate the posterior as a normal distribution with means θ_A^* and a diagonal precision matrix given by the diagonal of the Fisher information matrix F.

The loss function can thus be defined as

$$L(\theta) = L_B(\theta) + \sum_i \frac{\lambda}{2} F_i(\theta_i - \theta^*_{A,i})^2 \tag{6.3}$$

where $L(\theta)$ is the loss for both tasks A and B, $L_B(\theta)$ is the loss for task B, λ is a constant hyperparamter, $\theta^*_{A,i}$ are the optimised parameters for task A, F_i are the Fisher information parameters and θ_i are the parameters being optimised for task B after having been optimised for task A.

This can be contrasted with L_2 regularisation in that instead of constraining all the parameters equally using the term $\frac{\lambda}{2}\sum_i \theta_i^2$, the regularisation term only constrains those parameters relevant to task A. In addition, EWC constrains the parameters to θ^*, as opposed to 0.

When used for neural machine translation, $L_B(\theta)$ is the loss for the in-domain data only, λ defines how important the general domain data is compared with the in-domain data. and i indexes each parameter.

6.2 Configuration

The TensorFlow implementation of EWC was developed by Danielle Saunders of the Department of Engineering, University of Cambridge.

Firstly, the fisher information for each parameter was calculated in 200 steps. Furthermore, the weight of the penalty term on task B λ was set to 500, with a learning rate α of 0.002, as experimentally it has been shown that the model trains well when $\lambda \approx 1/\alpha$ [14]. Afterwards, the model was retrained on the biology training corpus for 20,000 iterations and then a further 20,000 steps. The results of these are shown in figures 6.2 and 6.1, respectively.

6.3 Results

As can be seen in figure 6.2, elastic weight improves the performance of the model on the biology test set from that of the biology-only model, when only 20,000 iterations are applied. The model now performs comparably to the health-then-biology model when decoding the biology test set. Furthermore, the degradation on the health domain is significantly less that that of the health-then-biology model. Thus, elastic weight consolidation is able to mitigate against catastrophic forgetting.

However, when elastic weight consolidation is compared to any of the interpolation techniques in Sections 4.2 or 5.3, these interpolation techniques can achieve a significantly

lower degradation on the health domain with a comaparable BLEU score on the biology domain. Nontheless, these techniques do require two models to be stored and run, as opposed to the single model for EWC. This would at least double the memory requirements and runtime of the NMT system. In the case of Bayesian interpolation, the worst-case runtime complexity could be $O(n^2)$ where n is the length of each target hypothesis under the modified implementation. On the other hand, that of the EWCNMT system would always be $O(n)$.

Thus, if the computational resources and time-constraints allow, it would be preferable to use an interpolation scheme. If not, EWC would provide a useful compromise between translation performance on the general-domain corpus and constraints on the storage and runtime.

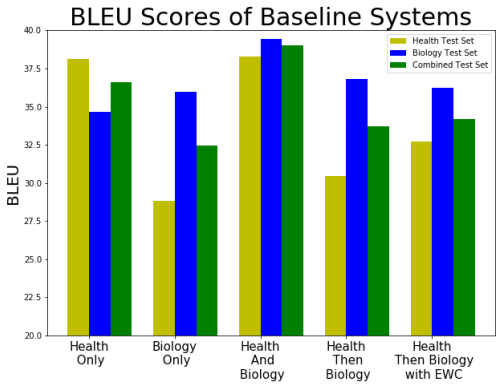


Fig. 6.1 Baseline experiments: the health-only, biology-only, health and biology, health then biology. The health then biology with EWC regularisation and 200000 iterations is included as well.

The model was run for a further 20,000 iterations on the biology test, and the performance on the biology test set was a noticeably better than that of health-then-biology without EWC. It is even better than most of the scores achieved with the interpolation experiments. However, the degradation on the health domain is even worse than that of the health-then-biology model, which suggests that the model no longer mitigates catastrophic forgetting. Hence, a total of 20,000 iterations should be enough to improve the performance on the specific-domain corpus, without too much degradation on the general-domain corpus.

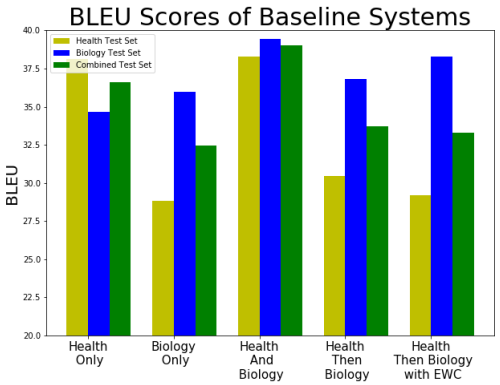


Fig. 6.2 Baseline experiments: the health-only, biology-only, health and biology, health then biology. The health then biology with EWC regularisation and 400000 iterations is included as well.

Chapter 7

Conclusion

7.1 Summary of Results

Catastrophic forgetting is a serious problem affecting many areas where neural models are applied, including machine translation. It has proven to be a significant obstacle to adapting models to specific domains without a substantial degradation of performance on the general domain. In order to understand how this could be achieved, several strategies, both at the decoding and training stages, were explored: fixed interpolation, Bayesian interpolation and elastic weight consolidation.

As can be seen from the baselines, it is clear that the performance on the biology set benefits from the contribution of the health corpus, as its performance increases when the biology training set is combined with that of health. On the other hand, there is no improvement of performance on the health test set when decoding using the mixed training set. Thus, any strategy should aim to find a suitable trade-off between the score on the biology test set and the score on the health test set.

Fixed interpolation demonstrated this trade-off, where the highest BLEU score on the biology test set was achieved when the health and biology models were equally weighted. Furthermore, reducing the weight on the biology model to 0.0 and increasing the weight on the health to 1.0 yielded the best BLEU score on the health test set. Nevertheless, when assigning weights between 0.5 and 1.0 on the health model and 0.0 and 0.5 on the biology model, the degradation in BLEU on the health test set was small compared with the improvement on that of biology.

Bayesian interpolation made use of the results of the fixed interpolation results in order to tune the values of the state-independent weights $\lambda_{k,t}$. It was also discovered that the trade-off in performance on the health and biology test sets was more biased towards that of the health, where λ was defined to be the identity matrix. As the ensemble performs best when the

biology and health models are equally weighted under fixed interpolation, $\lambda_{k,t}$ was set to 0.5 on both models when decoding the biology test set, which led to the best performance of the biology set so far.

Finally, elastic weight consolidation was shown to mitigate the degradation on the health domain after 20,000 iterations on the biology training set, albeit to a lesser degree than the interpolation models. Nonetheless the computational and and runtime requirements are significantly less on the model trained using EWC, which would mean that this NMT system could be deployed in situations where these are constrained. While training for a further 20,000 iterations improved the BLEU score on the biology test set, it worsened on the health test set.

7.2 Further Work

In the future, the source sentence could be used to predict the domain and thus determine the initialisation of $p(t)$ and $\lambda_{k,t}$. This could be done with the entropy values from maximum-entropy classifiers, or the scores from source LM classifiers introduced in [13]. These values would be used to select a class, but they could be used to calculate $p(t)$ using some form of mean.

Several classifiers can be trained on the source side of each parallel training corpus. These classifiers can be adapted to each domain by interpolating the classifiers and adjusting the interpolation weights to minimise the perplexity on the source sentence in the in-domain development. Thus, the domain of the source sentence can determined by selecting the domain-adapted LM with the maximum score. All of the domain-adapted classifiers must be interpolations of the same set of LMs.

Using a maximum-entropy classifer would allow the use of a greater number of features. These could include features from single words, pairs of adjacent words, the first or last words of a sentence. Furthermore, the source LM classifier defined above can be used to create indicator features, which indicate whether a given LM yields the maximum score.

Wang et al. [26], on the other hand devised a perceptron classifier which used domain-independent features for classification. The features involved in these experiments included domain word coverage, domain word coverage, domain word coverage by phrase length, average phrase length, domain phrase ratio and domain phrase ratio by phrase length. The features were designed on the assumption that if most of the source sentence words come from a given domain, the source sentence is likely to be from that domain. In addition, if several phrases retrieved from the source sentence are found in the training set for a given

domain, the source sentence is most probably from that domain. Again, the scores given to each task can be used to calculate $p(t)$.

References

[1] Allauzen, C. and Riley, M. (2011). Bayesian language model interpolation for mobile speech input. In *INTERSPEECH 2011, 12th Annual Conference of the International Speech Communication Association, Florence, Italy, August 27-31, 2011*, pages 1429–1432.

[2] Ballinger, B., Allauzen, C., Gruenstein, A., and Schalkwyk, J. (2010). On-demand language model interpolation for mobile speech input. In *INTERSPEECH*, pages 1812–1815. ISCA.

[3] Bartomeu, R. S. and Montraveta, A. M. F. (2012). *Syntactic mismatches between English and Spanish: a descriptive analysis and classification*. PhD thesis, Autonomous University of Barcelona.

[4] Bodén, M. (2001). A guide to recurrent neural networks and backpropagation.

[5] Bottou, L. (2010). Large-scale machine learning with stochastic gradient descent. In *in COMPSTAT*.

[6] Byrne, B. (2018). Lecture notes in statistical machine translation.

[7] Cho, K., van Merrienboer, B., Gülçehre, , Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. (2014). Learning phrase representations using rnn encoder-decoder for statistical machine translation. In Moschitti, A., Pang, B., and Daelemans, W., editors, *EMNLP*, pages 1724–1734. ACL.

[8] Chu, C. and Wang, R. (2018). A survey of domain adaptation for neural machine translation. *CoRR*, abs/1806.00258.

[9] Cohn, T., Schulz, P., and Aziz, W. (2018). A stochastic decoder for neural machine translation. In *ACL (1)*, pages 1243–1252. Association for Computational Linguistics.

[10] Dakwale, P. and Monz, C. (2017). Fine-tuning for neural machine translation with limited degradation across in- and out-of-domain data. In *MT 2017, Machine Translation Summit XVI Nagoya, Japan, September 18-27, 2017*, pages 156–169.

[11] Glorot, X., Bordes, A., and Bengio, Y. (2011). Deep sparse rectifier neural networks. In Gordon, G., Dunson, D., and Dudík, M., editors, *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, volume 15 of *Proceedings of Machine Learning Research*, pages 315–323, Fort Lauderdale, FL, USA. PMLR.

[12] Hinton, G. E., Vinyals, O., and Dean, J. (2015). Distilling the knowledge in a neural network. *CoRR*, abs/1503.02531.

[13] Huck, M., Birch, A., and Haddow, B. (2015). Mixed-Domain vs. Multi-Domain Statistical Machine Translation. In *Machine Translation Summit*, pages 240–255, Miami, FL, USA.

[14] Kirkpatrick, J., Pascanu, R., Rabinowitz, N. C., Veness, J., Desjardins, G., Rusu, A. A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A., Hassabis, D., Clopath, C., Kumaran, D., and Hadsell, R. (2016). Overcoming catastrophic forgetting in neural networks. *CoRR*, abs/1612.00796.

[15] Matsunaga, S. and Sakamoto, H. (1996). Two-pass strategy for continuous speech recognition with detection and transcription of unknown words. In *ICASSP*, pages 538–541. IEEE Computer Society.

[16] Neves, M. L., Jimeno-Yepes, A., and Névéol, A. (2016). The scielo corpus: a parallel corpus of scientific publications for biomedicine. In *LREC*. European Language Resources Association (ELRA).

[17] Ng, A. (2004). Feature selection, l1 vs. l2 regularization, and rotational invariance. In *Proceedings of the twenty-first international conference on Machine learning*, page 78. ACM.

[18] Papineni, K., Roukos, S., Ward, T., and Zhu, W.-J. (2002). Bleu: A method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, pages 311–318, Stroudsburg, PA, USA. Association for Computational Linguistics.

[19] Parisi, G. I., Kemker, R., Part, J. L., Kanan, C., and Wermter, S. (2018). Continual lifelong learning with neural networks: A review. *CoRR*, abs/1802.07569.

[20] Pellegrino, F., Coupé, C., and Marsico, E. (2011). A cross-language perspective on speech information rate. 87:539–558.

[21] Rusu, A. A., Rabinowitz, N. C., Desjardins, G., Soyer, H., Kirkpatrick, J., Kavukcuoglu, K., Pascanu, R., and Hadsell, R. (2016). Progressive neural networks. *CoRR*, abs/1606.04671.

[22] Stahlberg, F., Hasler, E., Saunders, D., and Byrne, W. (2017). SGNMT – a flexible NMT decoding platform for quick prototyping of new models and search strategies. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (Demo Papers)*. (5 pages).

[23] Sutskever, I., Vinyals, O., and Le, Q. V. (2014). Sequence to sequence learning with neural networks. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'14, pages 3104–3112, Cambridge, MA, USA. MIT Press.

[24] Vaswani, A., Bengio, S., Brevdo, E., Chollet, F., N. Gomez, A., Gouws, S., Jones, L., Kaiser, , Kalchbrenner, N., Parmar, N., Sepassi, R., Shazeer, N., and Uszkoreit, J. (2018). Tensor2tensor for neural machine translation.

[25] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. u., and Polosukhin, I. (2017). Attention is all you need. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc.

[26] Wu, H., Wang, H., and Zong, C. (2012). Improved domain adaptation for statistical machine translation. In *In Proceedings of the Conference of the Association for Machine translation, Americas*.

[27] Zhang, B., Xiong, D., and Su, J. (2017). A gru-gated attention model for neural machine translation. *CoRR*, abs/1704.08430.