A Comparison of Sentence-Weighting Techniques for NMT

Simon Riess simon@tiger-bytez.com
Matthias Huck mhuck@cis.uni-muenchen.de
Work done while the author was employed at
Center for Information and Language Processing, LMU Munich, Germany

Alexander Fraser fraser@cis.uni-muenchen.de
Center for Information and Language Processing, LMU Munich, Germany

Abstract

Sentence weighting is a simple and powerful domain adaptation technique. We carry out domain classification for computing sentence weights with 1) language model cross entropy difference 2) a convolutional neural network 3) a Recursive Neural Tensor Network. We compare these approaches with regard to domain classification accuracy, and study the posterior probability distributions. Then we carry out NMT experiments in the scenario where we have no in-domain parallel corpora, and only very limited in-domain monolingual corpora. Here, we use the domain classifier to reweight the sentences of our out-of-domain training corpus. This leads to improvements of up to 2.1 BLEU for German to English translation.

1 Introduction

Neural Machine Translation (NMT) outperforms phrase based SMT for settings with large amounts of parallel data. However, in general adding out-of-domain data during training does not particularly improve NMT translation quality and is sometimes even harmful. For SMT domain adaptation is well understood and can be classified into two main approaches: 1) model centric techniques adapt the training objective on instance level (e.g., sentence weighting or regularization) or model level (e.g., ensembling or language models), and 2) data centric techniques perform a sentence selection based on a score indicating the similarity between the sentence to be translated and in-domain data.

We combine ideas from model centric and data centric approaches. We apply CNNs and Recursive Neural Tensor Networks (RNTNs) to compute domain scores for sentence weighting in NMT. We compare with a Cross-Entropy classifier (XenC) as a well established baseline. Our approach modifies the training objective so that every sentence pair is scaled by its individual weight, with sentences most similar to the in-domain data having most impact during training.

Our classifier is trained on small amounts of in-domain and out-of-domain monolingual data. We then use the classifier to find useful sentences within the out-of-domain data, i.e., sentences which are similar to the in-domain data.

We carry out intrinsic (classification) and extrinsic (MT) experiments applying sentence classification for domain adaptation. The scores obtained by the CNN and RNTN are strongly peaked in comparison to the cross-entropy classifier, which is important for the NMT sentence weighting. As the neural classifiers showed rather extreme probability score distributions in the intrinsic experiments, we studied various transformations of the scores which we use to

find less peaked distributions. The resulting distributions showed less extreme behavior while preserving the strong classification ability. Applying our transformed scores to the task of sentence weighting for domain adaptation outperformed cross-entropy classifiers.

In summary, the contributions of this paper are as follows: 1) Neural classifiers show high confidence separating in- and out-of-domain data, higher than a cross-entropy classifier, hence posterior probabilities are distributed closely around the extremes 0 and 1. 2) The CNN and RNTN classifiers don't differ much from each other with respect to their score distributions, both are strongly peaked. 3) The extreme scores need to be transformed in order to be applied as weights in NMT, and we show how to do this effectively. 4) We show that using transformed CNN scores as weights during NMT training is better than a cross-entropy based classifier, which was the previous state-of-the-art solution.

2 Sentence-Weighting Techniques

In order to apply sentence weighting to the translation process, one first needs to come up with a method for scoring sentences with respect to how similar they are to in-domain data. Here we carry out a comparison between an established baseline (cross entropy) to the two different techniques based on neural networks that we have discussed (CNN and RNTN).

2.1 XenC: LM Cross-Entropy Difference

Language model (LM) cross-entropy difference scoring is a widely used technique for MT domain adaptation. The approach is implemented in the tool XenC (Rousseau, 2013). Here the difference between cross-entropy scores of sentences from the entire training corpus and the sentences of an in-domain corpus is computed. We applied monolingual cross-entropy difference as proposed by Moore and Lewis (2010), which is defined as

$$H(P_{LM}) = -\frac{1}{n}\sum_{i=1}^{n} log P_{LM}(w_i \mid w_i, \ldots, w_{i-1}) \quad (1)$$

where P_{LM} is the probability of the word w_i given the words w_1 to w_{i-1} for the language model LM. LM is estimated from the specified in-domain corpus. The formula is applied to all sentences in the training data for the NMT system, and is then interpreted as the sentence weight. XenC is not a neural system. It applies statistical computation of cross-entropy given an LM. The language model is a 4-gram model and Kneser-Ney smoothing is applied (Ney et al., 1994).

This approach is widely used throughout various papers and systems with regard to domain adaptation. It is mathematically relatively inexpensive and can therefore be computed very quickly even for extensive training corpora, without the need for GPU resources. These factors make it a suitable baseline for our comparisons to neural classification systems.

2.2 CNN Classifier

Convolutional neural networks (CNN) perform very well on tasks like image and sentence classification. In our case, we are classifying sentences in two classes, in-domain and out-of-domain. We applied a plain vanilla system by Yoon Kim (Kim, 2014), which consists of a simple CNN on top of pretrained word vectors. CNNs consist of layers with convolving filters learning local features. In this architecture one layer of convolution is applied on top of word vectors trained by Mikolov et al. (2013) on Google News. This approach performed well on several sentence classification tasks (Kim, 2014).

Figure 1 shows this simple model architecture. A sentence of length n (shorter sentences are padded) is represented as the concatenation of its word vectors. Similar to computer vision

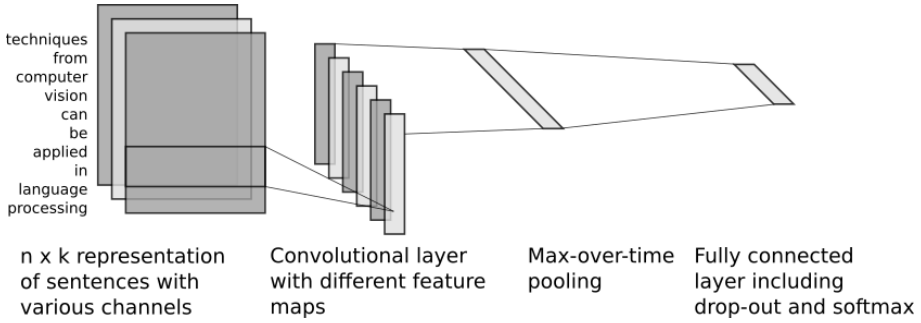


Figure 1: CNN model architecture.

tasks, filters are applied to words in a certain proximity to produce a new feature.

$$c_i = f(\mathbf{w} \cdot \mathbf{x}_{i:i+h-1} + b) \tag{2}$$

b is a bias term and f a non-linear activation function. The filter slides over the input sentence and therefore creates a feature map

$$\mathbf{c} = [c_1, c_2, \ldots, c_{n-h+1}] \tag{3}$$

Then max-over-time pooling is applied, $\hat{c} = max\{c\}$, to capture the most important feature for each feature map. Multiple filters are applied simultaneously and the max-pooling outputs form the penultimate layer. The last layer is a fully connected softmax layer to output the probability distribution over the labels.

For regularization to reduce over-fitting and improve generalization, Dropout and constraining the l_2-norms of weight vectors is applied (Krizhevsky et al., 2012). Dropout randomly drops out - i.e. setting to zero - a proportion p of hidden units (in this case in the last layer) during training. Given the output of the max-pooling layer $\mathbf{z} = [\hat{c_1}, \hat{c_2}, \ldots, \hat{c}_m]$, instead of

$$y = \mathbf{w} \cdot \mathbf{z} + b \tag{4}$$

dropout uses

$$y = \mathbf{w} \cdot (\mathbf{z} \circ \mathbf{r}) + b \tag{5}$$

with $\circ$ being element-wise multiplication and $\mathbf{r} \in \mathbb{R}^m$ a "masking" vector of bernoulli distributed random variables with probability p of being 1. Furthermore a threshold s for l_2-norms in introduced, rescaling $\mathbf{w}$ to $||\mathbf{w}||_2 = s$ if $||\mathbf{w}||_2 > s$ after a gradient descent step.

2.3 RNTN Classifier

CNNs work on word vectors and filters, which aggregate local information within a sentence. This is less expressive than richer forms of sentence representation, e.g., parse trees, which take into account the grammatical structure. To deal with parse trees for sentiment classification (Socher et al., 2013) introduced a recursive deep model, the Recursive Neural Tensor Network (RNTN).

The representations of sentences within recursive neural models apply to variable length and syntactic type and is used for classification. First, each sentence is parsed into a binary tree with leaf nodes being single words, represented by a vector. Then the parent vectors will be computed in a bottom-up fashion using compositionality functions g. The parent vectors themselves are recursively given as features to a classifier and their parents respectively.

Each word is represented by a d dimensional word vector. These are fed into activation functions and ultimately used in $softmax$ for classification.

Recursive Neural Network. The simplest approach is the standard recursive neural network (Goller and Küchler, 1996; Socher et al., 2011). First, the parents whose children are already computed (i.e. both children are words) will be evaluated with an activation function $f = tanh$. Following equations are used to evaluate the parent nodes according to Figure 2a:

$$p_1 = f\left(W\begin{bmatrix} b \\ c \end{bmatrix}\right), p_2 = f\left(W\begin{bmatrix} a \\ p_1 \end{bmatrix}\right) \tag{6}$$

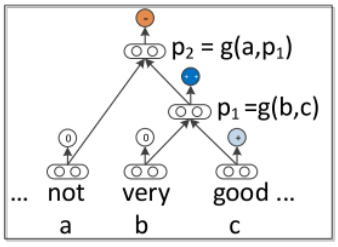
where $W \in \mathbb{R}^{dx2d}$ is the main learning parameter.

Matrix-Vector RNN. MV-RNNs are linguistically motivated in a sense that most of the parameters are linked with words and that the composition function depends on the actual words being combined. Each word and subphrase are represented as a vector and a matrix, which are combined in the composition function.

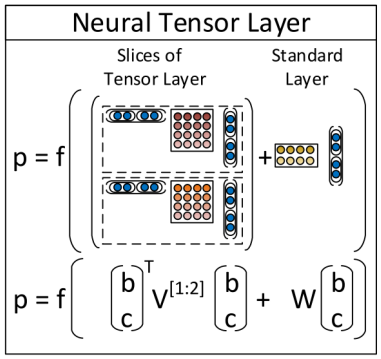
Each word's matrix initially is a dxd identity matrix with Gaussian noise. These matrices will be trained to optimise classification. Each sentence and subphrase is represented by a list of (vector, matrix) pairs and its parse tree. Following the same example from Figure 2a, the computation is as follows:

$$p_1 = f\left(W\begin{bmatrix} Cb \\ Bc \end{bmatrix}\right), P_1 = f\left(W_M\begin{bmatrix} B \\ C \end{bmatrix}\right), \tag{7}$$

while the parent pair (p_2, P_2) is computed using (p_1, P_1) and (a, A). The vectors are fed into the softmax function for classifying each subphrase.



(a) Recursive Neural Network: Parent vectors are computed in a bottom up fashion, with activation function g and node vectors as features for classification. Socher et al. (2013) page 4 (CC BY-NC-SA 3.0 license).



(b) A single layer of an RNTN: Representation of one of d-many slices, that can capture the type of influence a child node can have on its parents. Socher et al. (2013) page 6 (CC BY-NC-SA 3.0 license).

Figure 2: Recursive Neural Network and Recursive Neural Tree Network architecture

Recursive Neural Tensor Network. Since MV-RNNs combine vectors with matrices, the number of parameters becomes very large, also depending on vocabulary size. A fixed number of parameters would be more desirable. The standard recursive neural network has to be extended for this purpose, because there, different from the MV-RNN, the input vectors only interact with each other implicitly.

In search for a single, more powerful composition function to perform better and aggregate meaning from subphrases, they proposed the Recursive Neural Tensor Network. The output for a tensor product $h \in \mathbb{R}^d$ is computed as follows

$$h = \begin{bmatrix} b \\ c \end{bmatrix}^T V^{[1:d]} \begin{bmatrix} b \\ c \end{bmatrix}; h_i = \begin{bmatrix} b \\ c \end{bmatrix}^T V^i \begin{bmatrix} b \\ c \end{bmatrix}, \tag{8}$$

where $V^{[1:d]} \in \mathbb{R}^{2dx2dxd}$ is the tensor that defines multiple bilinear forms.

The RNTN uses a definition very similar to the standard recursive neural network for computing p_1:

$$p_1 = f\left(\begin{bmatrix} b \\ c \end{bmatrix}^T V^{[1:d]} \begin{bmatrix} b \\ c \end{bmatrix} + W \begin{bmatrix} b \\ c \end{bmatrix}\right) \tag{9}$$

The tensor V can directly relate input vectors and its slices can be interpreted as capturing specific types of composition, with a static number of parameters.

3 Intrinsic Evaluation: Domain Classification

3.1 Data

We study the interesting task of translation using limited in-domain monolingual corpora and larger out-of-domain parallel corpora, which is a realistic scenario. All classifiers were trained on 30k medical in-domain and 30k out-domain sentences, selected from the UFAL corpus.[1] This training data was the same for all three classifiers to allow comparison. The RNTN requires a certain input format, so the sentences were pre-processed by the Stanford Parser and brought into the necessary parse tree format.

For intrinsic evaluation, the classifiers were applied to gold standard test data. Newstest 2017 was used as out-of-domain data, whereas the medical HimL test set[2] was used as in-domain data. Both test sets contain about 2k sentences.

The trained classifiers were applied to the test sets, in the next section we analysed the classification errors and compared the respective probability score distribution.

3.2 Evaluation on Test sets

Figure 3 and Table 1 show the scoring outputs for in- and out-domain test data. These histograms indicate how many sentences in the test set where assigned a certain score with bins of width 0.05. An output of 1 means high confidence for in-domain data and 0 means high confidence for out-domain data.

Classifier	Acc. [%]	Out Acc. [%]	In Acc. [%]
CNN	80.4	87.9	72.8
RNTN	76.1	87.3	64.8
XenC	71.1	46.8	95.4

Table 1: Classification results on German out-of-domain and in-domain test data.

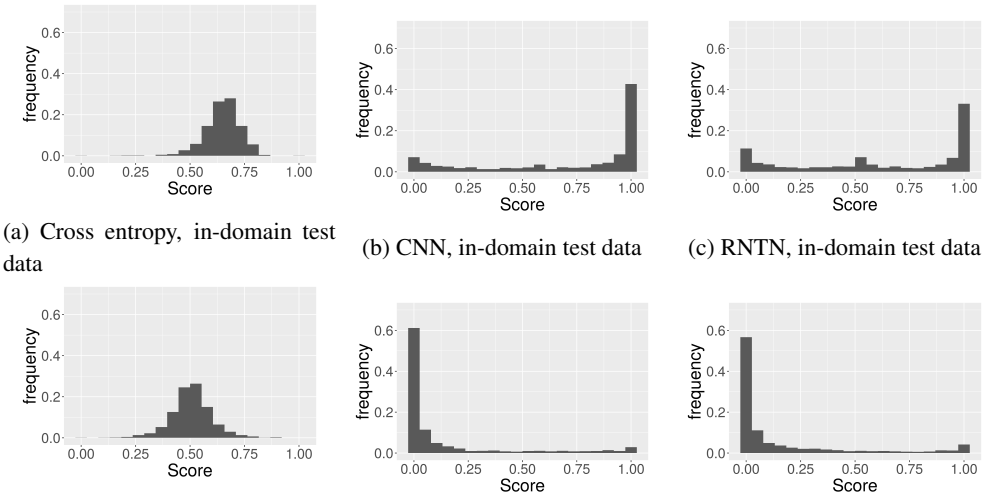
When comparing the results for the CNN and the RNTN, the differences are rather small, without obvious difference in shape of their distributions. We see a dominating peak at the correct side of the spectrum, which shows these classifiers have a high degree of confidence in their decisions. This peak diminishes rather quickly to then have a second minor peak around the other end of the spectrum.

This shape looks different for the cross entropy scoring. It resembles a bell curve with its mean slightly skewed towards the correct side of the spectrum. This shows a relatively unclear decision boundary between in- and out-of-domain data, since most of the sentences are scored rather in the middle between the two extremes.

[1] https://ufal.mff.cuni.cz/ufal_medical_corpus

[2] https://www.himl.eu/test-sets

These results should be taken with a grain of salt, as it is difficult to define pure in-domain and out-of-domain data. Discussions in the European Parliament (as found in the Europarl corpus) can revolve around medical topics, while being labeled as out-of-domain. Patient information as found in the data by the Health in my Language (HimL) project can include phrases of a more general nature, while being labeled in-domain. Such effects are not taken into account in our work.



(a) Cross entropy, in-domain test data

(b) CNN, in-domain test data

(c) RNTN, in-domain test data

(d) Cross entropy, out-of-domain test data

(e) CNN, out-of-domain test data

(f) RNTN, out-of-domain test data

Figure 3: Classifier outputs on German test data

3.3 Classifier probability scores on NMT training data

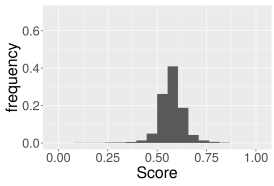
We applied the classifiers to the source (German) side of the NMT training data, leading to scores that can be used as weights during training the NMT system. Figure 4 shows the distribution of the scores for the CNN and the Cross Entropy classifier. Since we do have English data for the same 30K sentences, we also looked at this classification problem, but the graphs are very similar, so they are not presented. The similarity of English and German suggests that our work may apply well to other languages.

The scores by the XenC classifier look similar to a normal distribution, with its mean around 0.5-0.6. Most of the sentences are scored with similar values, indicating an average importance during learning. There are few outliers, overall the distribution is rather narrow with a low standard deviation.

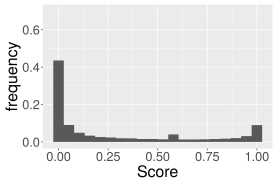
The scores by the CNN classifier look significantly different. Instead of the expected normal distribution, most of the weights are below 0.1 with a few scores above 0.95. This means that the classifier is very confident in it's decisions. This high level of confidence is also visible in Figure 3.

4 Extrinsic Evaluation: Neural Machine Translation

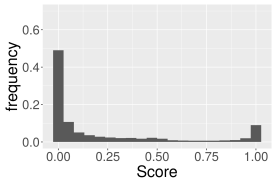
In this section we first present our score transformations, and then we present the experiments and results.



(a) Cross Entropy classifier XenC applied to NMT training data



(b) CNN classifier applied to NMT training data



(c) RNTN classifier applied to NMT training data

Figure 4: Classifier outputs on German NMT training data

4.1 Score Transformations

In initial experiments (which we present in detail later), we found that without applying score transformations instance weighting training of NMT models does not converge. During sentence weighting, the probability score from the classifiers is multiplied with the learning rate. As mentioned previously, the high classification confidence in neural classifiers lead to a vast majority of sentences scored very close to 0, setting the learning rate during training very low. This restricts the Transformer to only learn fully on a small subset of its original training data. We suppose the rather extreme original probability scores let the NMT starve for data.

For the purpose of sentence weighting, the data distributions from the classifier outputs are problematic in a sense that they put most of the mass to the borders of the distribution, i.e., almost all of the scores are very close to 0 or 1. This impacts the sentence weighting techniques significantly, since a score that is almost 0 effectively excludes these sentences from the data set. We therefore applied several score transformations to obtain a normalized score distribution, as we describe next.

Parabolic Transformation. The first approach is to multiply each of the scores with a linear function to increase the very low scores and decrease the very high scores. Here we chose a simple linear function by taking an educated guess without doing further hyperparameter optimisation. For every score x we applied the function

$$f(x) = x * (-4.2 * x + 5) \tag{10}$$

which results in a parabola with its peak around $x = 0.5$. A parabola in this shape increases low scores and decreases high scores. Its parameters were an educated guess, leading to competitive results in preliminary experiments.

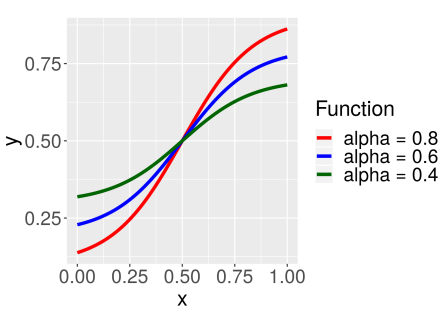
Sigmoidal Transformation. The second approach is to limit the scores into a certain interval using a sigmoid function. We tried different hyperparameters indicating different intervals according the following function

$$\alpha * 1/(1 + exp(-6 * (x - 0.5))) + (1 - \alpha)/2 \tag{11}$$

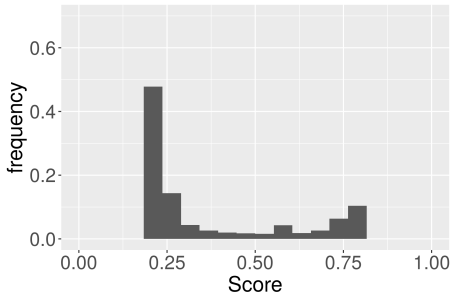
indicating the interval $[0.5 - \alpha/2, 0.5 + \alpha/2]$. These functions are shown in Figure 5a, leading to a normalised distribution on the NMT training data shown in Figure 5b.

Quantile Transformation. The previous approaches lead to narrower and flatter data distributions. As a third approach, we made the distribution completely uniform.

The second attempt was to "normalise" the quantiles by considering the negatively classified (0-0.5) and the positive (0.5-1) sentences separately and then performing the quantile transformation on both subsets individually. Both categories were transformed into quantiles according to their own distribution and then transformed back into the respective interval.



(a) Plot of sigmoidal transformation.



(b) CNN weights for NMT training data after sigmoidal transformation with $\alpha = 0.6$

Figure 5: Sigmoidal transformation and its effects on the probability score distribution

4.2 Experiments and Results

For our translation experiments we applied Marian (Junczys-Dowmunt et al., 2018) because of its ability to incorporate sentence weighting. It offers a transformer (Vaswani et al., 2017) implementation that closely follows the original architecture. This setup is shown to achieve state-of-the-art results. Marian is C++ based, which makes it very time efficient.

We assume a scenario with a sufficient amount of parallel out-of-domain data, but only a small amount of monolingual in-domain data on the source side. We use the classifiers we trained before. 3M out-of-domain sentences (of which 2M are from Europarl, see the UFAL corpus web page) from the UFAL corpus are used for training NMT. We report on two well-known MT test sets (Cochrane and NHS24) which are both from the medical domain.

Table 2 gives an overview of all performed experiments. A baseline transformer model (Table 2, row 1) was trained without any domain specific adaptation.

Since we assume we have 30K of monolingual in-domain data, we wanted to evaluate whether giving the NMT system access to this data could be effective. Since we had a translation of this 30K available, we actually fine-tuned on parallel data (i.e., we assumed perfect translation of the 30K, so this is an upper bound of the gains that could be obtained). The results (row 2) show that this is too little data to make much of a difference in translation quality (0.2 to 0.4 BLEU gains), which is not surprising given the very large out-of-domain corpus. The strong results we present below are qualitatively different from having access to a small amount of in-domain data to train on (even small amounts of in-domain parallel data).

The results for the the XenC classifier (row 3) serve as a stronger baseline for our results with the neural classifiers. We also tried to directly apply the scores from the neural classifiers, but this led to bad or unstable models that did not coverge (not shown in table). Too many sentences are scored too close to 0, letting their impact vanish, not allowing the training to converge. As discussed earlier and shown in Figure 4 for the CNN, most of the probability mass of the CNN's score distribution is concentrated at the extremes, 0 and 1, leading to many sentences having nearly no impact during training (this is similar for the RNTN as well). This is similar to training with too little data, as weighting a sentence very close to 0 skips the sentence.

These effects can be repaired by adding +1 to the classifier scores (rows 4-6), leading to improvements over the baseline for all trained systems, especially for the two neural classifiers. Further experiments focused on the CNN because it outperforms the RNTN and is simpler.

Following this we looked at score transformations. The scores from the CNN were manipulated by various sigmoidal transformations (rows 7-9), as its results in the first experiments looked most promising. As the qualitative analysis already showed in Figure 5b, after the sigmoidal transformation the CNN scores look more natural. The experiment results indicate that

this transformation also lead to major improvements (rows 7-9), producing the best result (row 8) among our experiments, an improvement over the baseline of 2.1 BLEU. The sigmoid transformation keeps the CNN's ability to clearly distinguish between in-domain and out-domain sentences from the test sets - much clearer than XenC.

After analysing the results of different values for α on the score distribution for the training data, we restricted our hyperparameter search to three values, covering a reasonably big range, without requiring an excessive number of NMT training runs, which was not possible given our resources. $\alpha = 0.6$ seemed promising as higher values barely change the score distribution and lower values result in very narrow distributions, and indeed leads to better NMT results.

Another possibility of combining the CNN's classifying power and the XenC's natural score distribution, is averaging their scores (rows 10,11). This also lead to improvements over the baseline but could not beat the CNN in combination with the sigmoidal transformation (row 8).

	MT System		BLEU	
			NHS24	**Cochrane**
(1)	Baseline		24.2	24.5
(2)	+ Fine-tuning		24.6	24.7
	Weighting	**With Transformation**		
(3)	XenC		23.2	23.8
(4)	XenC	+1	24.2	25.2
(5)	RNTN	+1	24.7	25.2
(6)	CNN	+1	24.9	25.7
(7)	CNN	$\text{Sigmoidal}_{0.8}$	24.7	25.7
(8)	CNN	$\text{Sigmoidal}_{0.6}$	**25.3**	**26.6**
(9)	CNN	$\text{Sigmoidal}_{0.4}$	24.6	25.7
(10)	CNN + XenC		24.9	25.7
(11)	CNN + XenC	+1	25.2	25.5
(12)	CNN	Parabolic +1	24.7	25.2
(13)	CNN	$\text{Sigmoidal}_{0.8}$ +1	24.4	25.8
(14)	CNN	$\text{Sigmoidal}_{0.6}$ +1	24.6	25.7
(15)	CNN	$\text{Sigmoidal}_{0.4}$ +1	24.1	25.8
(16)	CNN	Quantiles 10 +1	24.8	25.7
(17)	CNN	Quantiles NegPos +1	24.5	25.5

Table 2: Machine translation quality. We report case-sensitive BLEU of postprocessed translations.

Finally, as adding +1 to the scores improved the results for all classifiers, we also applied +1 to the previously described transformations (rows 12-17). This still lead to minor improvements over the baseline system, but was harmful to the CNN and its sigmoidal transformation.

In summary we saw that classifier outputs might be too extreme in their distribution, which can be normalised by transformations to even outperform baseline approaches. Neural classifiers show stronger abilities to distinguish between in-domain and out-of-domain data than cross-entropy based classifiers, resulting in higher BLEU scores when applied in sentence weighting.

5 Related Work

Domain adaptation strategies can be separated into four categories: data selection, data generation, instance weighting and model interpolation (Chu and Wang, 2018). We focus our discussion on data selection and instance weighting, as these are closely related to our approach.

Data-centric methods. Models are trained using in-domain and out-of-domain data to evaluate out-of-domain data and compute a similarity score. Using a cut-off threshold on these scores the training data can be selected. Language Models (Moore and Lewis, 2010; Axelrod et al., 2011; Duh et al., 2013) or joint models (Cuong and Sima'an, 2014; Durrani et al., 2015) can traditionally be applied to score corpora. Recently convolutional neural networks (CNN)

(Chen et al., 2016) were used. Our work has similarities to this work but uses instance weighting rather than data selection.

In settings where the amount of parallel training corpora is not sufficient, generating pseudo-parallel sentences by information retrieval (Utiyama and Isahara, 2003), self-enhancing (Lambert et al., 2011) or parallel word embeddings (Marie and Fujita, 2017). Aside from generating sentences, other approaches generate monolingual n-grams (Wang et al., 2014) or parallel phrase pairs (Chu, 2015).

In general, data-centric methods (data selection and data generation) are not SMT specific and can be directly applied to NMT. However, because these methods are not directly related to NMT's training criterion, they only lead to minor improvements (Wang et al., 2017a).

Model-centric methods. Instance Weighting is a technique from SMT and was introduced to NMT as well (Wang et al., 2017b). An in-domain language model was trained to measure the similarity between sentences and the in-domain data via cross-entropy. The weights are then integrated into the training objective. We improve on their work by using state-of-the-art neural classifiers and showing that they are more effective than cross-entropy.

Two works that are closer to our work are those of Wang et al. (2018) and Chen et al. (2017). Wang et al. (2018) generates sentence embeddings for all in-domain sentences and then measures the distance between every sentence and the in-domain core. The underlying assumption is that the core of all in-domain sentence embeddings is a typical representative and proximity in their sentence embeddings indicates being part of the same domain. This approach is appropriate when we have in-domain parallel text, but we study a different scenario, with no access to in-domain parallel text, which means the encoder has no access to in-domain training examples. In Chen et al. (2017) a domain classifier is incorporated into the NMT system, using features from the encoder to distinguish between in-domain and out-of-domain data. The classifier probabilities are used to weight sentences with regard to their similarity to in-domain data, when training the neural network. Scaling the loss function is similar to multiplying the learning rate with the instance weight. The classifier and NMT are trained at the same time, whereas we chose an approach with pretrained neural classifiers which are trained on a small amount of monolingual data (the scenario we study) with no access to parallel in-domain data.

Finally, while some previous work we have mentioned did look at various ways to use domain classification, such previous work has not focused on how to weight the classifier probabilities for effective use in NMT, which we showed is important for obtaining translation quality improvements, particularly when using neural classifiers which can be overconfident.

6 Conclusion

Neural classifiers have high confidence when separating in-domain from out-of-domain data, leading to a strong decision boundary. Classification results are good, but the boundary was too drastic, resulting in a poor score distribution with most mass near 0 and 1. This can be fixed by adding +1, keeping sentences with a low score as they are and giving a bonus to sentences with a higher score. The scores from, e.g., a CNN, can be transformed by a sigmoid function, making the score distribution more natural while keeping its strong decision boundary. Cross-entropy approaches lead to a poor score distribution. Sigmoid CNN scores performed best.

Our MT experiments showed that neural classifiers can be used to score out-of-domain data effectively. Our work showed that simple transformations of classifier outputs are necessary. The use of the transformed scores by applying sentence weighting on the NMT training data improves translation quality. Our research shows that results from CNNs trained on domain classification achieve significant domain adaptation effects in NMT. It was important to carry out light-weight score transformations. We outperformed baseline experiments by up to 2.1 BLEU points.

Acknowledgments

This project has received funding from the European Research Council under the European Union's Horizon 2020 research and innovation program (grant agreement #640550).

References

Axelrod, A., He, X., and Gao, J. (2011). Domain adaptation via pseudo in-domain data selection. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '11, pages 355–362, Stroudsburg, PA, USA.

Chen, B., Cherry, C., Foster, G. F., and Larkin, S. (2017). Cost weighting for neural machine translation domain adaptation. In *NMT@ACL*.

Chen, B., Kuhn, R., Foster, G., Cherry, C., and Huang, F. (2016). Bilingual methods for adaptive training data selection for machine translation. In *AMTA*.

Chu, C. (2015). Integrated parallel data extraction from comparable corpora for statistical machine translation. Doctoral Thesis, Kyoto University.

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

Cuong, H. and Sima'an, K. (2014). Latent domain translation models in mix-of-domains haystack. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 1928–1939, Dublin, Ireland.

Duh, K., Neubig, G., Sudoh, K., and Tsukada, H. (2013). Adaptation data selection using neural language models: Experiments in machine translation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 678–683, Sofia, Bulgaria.

Durrani, N., Sajjad, H., Joty, S., Abdelali, A., and Vogel, S. (2015). Using joint models for domain adaptation in statistical machine translation.

Goller, C. and Küchler, A. (1996). Learning task-dependent distributed representations by backpropagation through structure. In *In Proc. of the ICNN-96*, pages 347–352.

Junczys-Dowmunt, M., Grundkiewicz, R., Dwojak, T., Hoang, H., Heafield, K., Neckermann, T., Seide, F., Germann, U., Fikri Aji, A., Bogoychev, N., Martins, A. F. T., and Birch, A. (2018). Marian: Fast neural machine translation in C++. In *Proceedings of ACL 2018, System Demonstrations*, pages 116–121, Melbourne, Australia.

Kim, Y. (2014). Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1746–1751.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In Pereira, F., Burges, C. J. C., Bottou, L., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105.

Lambert, P., Schwenk, H., Servan, C., and Abdul-Rauf, S. (2011). Investigations on translation model adaptation using monolingual data. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, WMT '11, pages 284–293, Stroudsburg, PA, USA.

Marie, B. and Fujita, A. (2017). Efficient extraction of pseudo-parallel sentences from raw monolingual data using word embeddings. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 392–398, Vancouver, Canada.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. (2013). Distributed representations of words and phrases and their compositionality. In Burges, C. J. C., Bottou, L., Welling, M., Ghahramani, Z., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems 26*, pages 3111–3119.

Moore, R. C. and Lewis, W. (2010). Intelligent selection of language model training data. In *Proceedings of the ACL 2010 Conference Short Papers*, pages 220–224, Uppsala, Sweden.

Ney, H., Essen, U., and Kneser, R. (1994). On structuring probabilistic dependencies in stochastic language modelling. *Computer Speech and Language*, 8:1–38.

Rousseau, A. (2013). Xenc: An open-source tool for data selection in natural language processing. *The Prague Bulletin of Mathematical Linguistics*, 100:73–82.

Socher, R., Lin, C. C.-Y., Ng, A. Y., and Manning, C. D. (2011). Parsing natural scenes and natural language with recursive neural networks. In *Proceedings of the 28th International Conference on International Conference on Machine Learning*, ICML'11, pages 129–136, USA.

Socher, R., Perelygin, A., Wu, J., Chuang, J., Manning, C. D., Ng, A., and Potts, C. (2013). Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Seattle, Washington, USA.

Utiyama, M. and Isahara, H. (2003). Reliable measures for aligning japanese-english news articles and sentences. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics - Volume 1*, ACL '03, pages 72–79, Stroudsburg, PA, USA.

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.

Wang, R., Finch, A., Utiyama, M., and Sumita, E. (2017a). Sentence embedding for neural machine translation domain adaptation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 560–566, Vancouver, Canada.

Wang, R., Utiyama, M., Finch, A. M., Liu, L., Chen, K., and Sumita, E. (2018). Sentence selection and weighting for neural machine translation domain adaptation. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 26:1727–1741.

Wang, R., Utiyama, M., Liu, L., Chen, K., and Sumita, E. (2017b). Instance weighting for neural machine translation domain adaptation. In *EMNLP*.

Wang, R., Zhao, H., Lu, B.-L., Utiyama, M., and Sumita, E. (2014). Neural network based bilingual language model growing for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 189–195, Doha, Qatar.