BERTTune: Fine-Tuning Neural Machine Translation with BERTScore

Inigo Jauregi Unanue[1,2], Jacob Parnell[1,2], Massimo Piccardi[1]
[1]University of Technology Sydney, NSW 2007, Australia
[2]RoZetta Technology, NSW 2000, Australia
Inigo.Jauregi@rozettatechnology.com
Jacob.Parnell@rozettatechnology.com
Massimo.Piccardi@uts.edu.au

Abstract

Neural machine translation models are often biased toward the limited translation references seen during training. To amend this form of overfitting, in this paper we propose fine-tuning the models with a novel training objective based on the recently-proposed BERTScore evaluation metric. BERTScore is a scoring function based on contextual embeddings that overcomes the typical limitations of n-gram-based metrics (e.g. synonyms, paraphrases), allowing translations that are different from the references, yet close in the contextual embedding space, to be treated as substantially correct. To be able to use BERTScore as a training objective, we propose three approaches for generating *soft predictions*, allowing the network to remain completely differentiable end-to-end. Experiments carried out over four, diverse language pairs have achieved improvements of up to 0.58 pp (3.28%) in BLEU score and up to 0.76 pp (0.98%) in BERTScore (F_{BERT}) when fine-tuning a strong baseline.

1 Introduction

Neural machine translation (NMT) has imposed itself as the most performing approach for automatic translation in a large variety of cases (Sutskever et al., 2014; Vaswani et al., 2017). However, NMT models suffer from well-known limitations such as overfitting and moderate generalization, particularly when the training data are limited (Koehn and Knowles, 2017). This mainly stems from the fact that NMT models have large capacity and are usually trained to maximize the likelihood of just a single reference sentence per source sentence, thus ignoring possible variations within the translation (e.g. synonyms, paraphrases) and potentially resulting in overfitting. A somewhat analogous problem affects evaluation, where metrics such as BLEU (Papineni et al., 2002) only consider as correct the predicted n-grams that match exactly in the ground-truth sentence. In order to alleviate the n-gram matching issue during evaluation, Zhang et al. (2020) have recently proposed the BERTScore metric that measures the accuracy of a translation model in a contextual embedding space. In BERTScore, a pretrained language model (e.g. BERT (Devlin et al., 2019)) is first used to compute the contextual embeddings of the predicted sentence, $\langle \hat{\mathbf{y}}_1, \ldots, \hat{\mathbf{y}}_k \rangle$, and the reference sentence, $\langle \mathbf{y}_1, \ldots, \mathbf{y}_l \rangle$, with k and l word-pieces, respectively. Then, recall (R_{BERT}), precision (P_{BERT}), and F1 (F_{BERT}) scores are defined as cosine similarities between the normalized contextual embeddings. For example, the recall is defined as:

$$R_{BERT} = \frac{1}{|l|} \sum_{y_i \in y} \max_{\hat{y}_j \in \hat{y}} \mathbf{y}_i^T \hat{\mathbf{y}}_j \qquad (1)$$

where the $\max$ function acts as an alignment between each word in the reference sentence (y) and the words in the predicted sentence ($\hat{y}$). Conversely, P_{BERT} aligns each word of the predicted sentence with the words of the reference sentence, and F_{BERT} is the usual geometric mean of precision and recall. Note that with this scoring function a candidate and reference sentences with similar embeddings will be assigned a high score even if they differ completely in terms of categorical words. Zhang et al. (2020) have shown that this evaluation metric has very high correlation with the human judgment.

In this work, we propose using BERTScore as an objective function for model fine-tuning. Our rationale is that BERTScore is a sentence-level objective that may be able to refine the performance of NMT models trained with the conventional, token-level log-likelihood. However, in order to fine-tune the model with BERTScore as an objective, end-to-end differentiability needs to be ensured. While the BERTScore scoring function is based on word embeddings and is in itself differentiable, its input derives from categorical predictions (i.e. argmax or sampling), breaking the differentiability of the overall model. In this work, we solve this problem by generating *soft predictions* during training with three different approaches. One of the approaches, based on the Gumbel-Softmax (Jang et al., 2017), also leverages sampling, allowing the model to benefit from a certain degree of *exploration*. For immediacy, we refer to our approach as *BERTTune*. The experimental results over four, diverse language pairs have shown improvements of up to 0.58 pp (3.28%) in BLEU score and up to 0.76 pp (0.98%) in BERTScore with respect to a contemporary baseline (Ott et al., 2019).

2 Related Work

In recent years, various researchers have addressed the problem of overfitting in NMT models. This problem can be specially severe for neural models, given that, in principle, their large number of parameters could allow for a perfect memorization of the training set. For instance, Ma et al. (2018) have trained an NMT model using both a reference sentence and its bag-of-words vector as targets, assuming that the space of alternative, correct translations share similar bags-of-words. Others (Elbayad et al., 2018; Chousa et al., 2018) have proposed smoothing the probability distribution generated by the decoder using the embedding distance between the predicted and target words, forcing the network to increase the probability of words other than the reference. Another line of work has proposed to explicitly predict word embeddings, using the cosine similarity with the target embedding as the reward function (Kumar and Tsvetkov, 2019; Jauregi Unanue et al., 2019).

Reinforcement learning-style training has also been used to alleviate overfitting (Ranzato et al., 2016; Edunov et al., 2018). The use of beam search removes the exposure bias problem (Wiseman and Rush, 2016), and the use of sampling introduces some degree of exploration. In addition, these approaches allow using non-differentiable, sequence-level metrics as reward functions. However, in practice, approximating the expectation of the objective function with only one or a few samples results in models with high variance and convergence issues.

Significant effort has also been recently dedicated to leveraging large, pretrained language models (Devlin et al., 2019; Radford et al., 2018; Peters et al., 2018) for improving the performance of NMT models. This includes using contextual word embeddings either as input features (Edunov et al., 2019) or for input augmentation (Yang et al., 2020; Zhu et al., 2020), and using a pretrained language model for initializing the weights of the encoder (Clinchant et al., 2019). Alternatively, Baziotis et al. (2020) have proposed using a pretrained language model as a prior, encouraging the network to generate probability distributions that have a high likelihood in the language model. In abstractive summarization, Li et al. (2019) have used BERTScore as reward in a deep reinforcement learning framework. In a similar vein, our work, too, aims to leverage pretrained language models for improving the NMT accuracy. However, to the best of our knowledge, ours is the first work to directly include a language model as a differentiable evaluation measure in the training objective. In this way, the NMT model is able to exploit the value of a pretrained language model while at the same time being fine-tuned over a task-specific evaluation metric.

3 BERTScore Optimization

Translation evaluation metrics, including BERTScore, typically require a predicted translation, $\langle \hat{y}_1, \ldots, \hat{y}_k \rangle$, and at least one reference translation, $\langle y_1, \ldots, y_l \rangle$, as inputs. At its turn, the predicted translation is typically obtained as a sequence of individual word (or token) predictions, using beam search or greedy decoding. We can express the predictions as:

$$\hat{y}_j = \arg\max_y p(y|x, \hat{y}_{j-1}, \theta) \quad j = 1, \ldots, k \tag{2}$$

where x represents the source sentence and θ the model's parameters. During model training, it is common practice to use teacher forcing (i.e., use words from the reference sentence as $\hat{y}_{j-1}$) for efficiency and faster convergence.

In brief, the computation of BERTScore works as follows: the scorer first converts the words in the predicted and reference sentences to corresponding static (i.e., non-contextual) word embeddings using the embedding matrix, $\mathbf{E}$, stored in the pretrained language model. For the predicted sequence, we note this lookup as:

$$\mathbf{e}_{\hat{y}_j} = \text{emb}_{LM}(\mathbf{E}, \hat{y}_j) \quad j = 1, \ldots, k \tag{3}$$

The sequences of static embeddings for the predicted and reference sentences are then used as inputs into the language model to generate corresponding sequences of contextualized embeddings, $\langle \hat{\mathbf{y}}_1, \dots, \hat{\mathbf{y}}_k \rangle$ and $\langle \mathbf{y}_1, \dots, \mathbf{y}_k \rangle$, respectively, over which the BERTScore is finally computed. For our work, we have chosen to optimize the F_{BERT} score as it balances precision and recall. For more details on the scoring function we refer the reader to (Zhang et al., 2020).

3.1 Soft predictions

However, it is not possible to directly use the F_{BERT} score as a training objective since the argmax function in (2) is discontinuous. Therefore, in this work we propose replacing the hard decision of the argmax with "soft predictions" that retain differentiability. Let us note concisely the probability in (2) as p_j^i, where i indexes a particular word in the V-sized vocabulary and j refers to the decoding step, and the entire probability vector at step j as $\mathbf{p}_j$. Let us also note as $\mathbf{e}^i$ the embedding of the i-th word in the embedding matrix of the pretrained language model, $\mathbf{E}$. We then compute an "expected embedding" as follows:

$$\bar{\mathbf{e}}_{\hat{y}_j} = \mathbb{E}[\mathbf{E}]_{\mathbf{p}_j} = \sum_{i=1}^{V} p_j^i \mathbf{e}^i \tag{4}$$

In other terms, probabilities $\mathbf{p}_j$ act as attention weights over the word embeddings in matrix $\mathbf{E}$, and the resulting expected embedding, $\bar{\mathbf{e}}_{\hat{y}_j}$, can be seen as a trade-off, or weighted average, between the embeddings of the words with highest probability. To be able to compute this expectation, the NMT model must share the same target vocabulary as the pretrained language model. Once the expected embeddings for the whole predicted sentence, $\langle \bar{\mathbf{e}}_{\hat{y}_1}, \dots, \bar{\mathbf{e}}_{\hat{y}_k} \rangle$, are computed, they are input into the language model to obtain the corresponding sequence of predicted contextualized embeddings, and the F_{BERT} score is computed. The fine-tuning loss is simply set as $\mathcal{L} = -F_{BERT}$. During fine-tuning, only the parameters of the NMT model are optimized while those of the pretrained language model are kept unchanged.

3.2 Sparse soft predictions

A potential limitation of using the probability vectors to obtain the expected embeddings is that they are, a priori, dense, with several words in the vocabulary possibly receiving a probability significantly higher than zero. In this case, the expected embeddings risk losing a clear interpretation. While we could simply employ a softmax with temperature to sparsify the probability vectors, we propose exploring two more contemporary approaches:

- **Sparsemax** (Martins and Astudillo, 2016): Sparsemax generates a Euclidean projection of the logits computed by the decoder (noted as vector $\mathbf{s}_j$) onto the probability simplex, Δ^{V-1}:

  $$\mathbf{p}_j^{SM} = \underset{\mathbf{p}_j \in \Delta^{V-1}}{\arg\min} ||\mathbf{p}_j - \mathbf{s}_j||^2 \tag{5}$$

  The larger the logits, the more likely it is that the resulting $\mathbf{p}_j^{SM}$ vector will have a large number of components equal to zero. The sparsemax operator is fully differentiable.

- **Gumbel-Softmax** (Jang et al., 2017; Maddison et al., 2017): The Gumbel-Softmax is a recent reparametrization technique that allows sampling *soft* categorical variables by transforming samples of a Gumbel distribution. The transformation includes a temperature parameter, τ, that allows making the resulting soft variables more or less sparse. By noting a sample from the Gumbel distribution as g^i, the Gumbel-Softmax can be expressed as:

  $$p_j^{i\,GS} = \frac{\exp((\log p_j^i + g^i)/\tau)}{\sum_{v=1}^{V} \exp((\log p_j^v + g^v)/\tau)} \tag{6}$$

  where $p_j^{i\,GS}$, $i = 1, \dots, V$, are the components of the probability vector used in (4). In the experiments, τ has been set to 0.1 to enforce sparsity. In addition to obtaining more "selective" predictions, the Gumbel-Softmax leverages sampling, allowing the fine-tuning to avail of a certain degree of *exploration*. The Gumbel-Softmax, too, is fully differentiable.

4 Experiments

4.1 Datasets

We have carried out multiple experiments over four, diverse language pairs, namely, German-English (de-en), Chinese-English (zh-en), English-Turkish (en-tr) and English-Spanish (en-es), using the datasets from the well-known IWSLT 2014 shared task[1], with 152K, 156K, 141K and 172K training sentences, respectively. Following Edunov et al. (2018), in the de-en dataset we have used $7,000$ samples of the training data for validation, and *tst2010*, *tst2011*, *tst2012*, *dev2010* and

[1] https://wit3.fbk.eu/2014-01

Model	**de-en**			**zh-en**			**en-tr**			**en-es**		
	BLEU	F_{BERT}	MS	BLEU	F_{BERT}	MS	BLEU	F_{BERT}	MS	BLEU	F_{BERT}	MS
Transformer NMT	33.61	77.56	52.86	18.28	68.04	34.81	17.68	76.55	18.3	37.80	79.31	45.76
+ BERTTune (DV)	33.58	77.90	53.4†	**18.53**†	**68.53**†	**35.57**†	17.81†	76.57	18.19	37.36	79.30	**45.92**†
+ BERTTune (SM)	33.39	77.88	53.27†	18.09	68.48†	35.18†	17.52	76.55	18.09	37.70	79.27	45.89†
+ BERTTune (GS)	**33.97**	**78.32**†	**53.58**†	18.39†	68.45†	35.33†	**18.26**†	**76.75**†	**18.33**	**37.96**†	**79.33**	45.84†

Table 1: Average BLEU, F_{BERT} and MoverScore (MS) results over the test sets. (†) refers to statistically significant differences with respect to the baseline computed with a bootstrap significance test with a p-value < 0.01 (Dror et al., 2018). The bootstrap test was carried out at sentence level for F_{BERT} and MS, and at corpus level for BLEU.

dev2012 as the test set. For the other language pairs, we have used the validation and test sets provided by the shared task. More details about the preprocessing are given in Appendix A.

4.2 Models and training

We have implemented the fine-tuning objective using the *fairseq* translation toolkit[2] (Ott et al., 2019). The pretrained language models for each language have been downloaded from Hugging Face (Wolf et al., 2020)[3]. As baseline, we have trained a full NMT transformer until convergence on the validation set. With this model, we have been able to reproduce or exceed the challenging baselines used in (Zhang et al., 2020; Xia et al., 2019; Miculicich et al., 2018; Wu et al., 2020). The fine-tuning with the F_{BERT} loss has been carried out over the trained baseline model, again until convergence on the validation set. For efficient training, we have used teacher forcing in all our models. During inference, we have used beam search with beam size 5 and length penalty 1. As performance measures, we report the BLEU, F_{BERT} and MoverScore (MS) (Zhao et al., 2019) results over the test sets averaged over three independent runs. Including BLEU and MS in the evaluation allows us to probe the models on metrics different from that used for training. Similarly to F_{BERT}, MS, too, is a contextual embedding distance-based metric, but it leverages soft alignments (many-to-one) rather than hard alignments between words in the candidate and reference sentences. To make the evaluation more probing, for MS we have used different pretrained language models from those used with F_{BERT}. For more details on the models and hyperparameter selection, please refer to Appendix A.

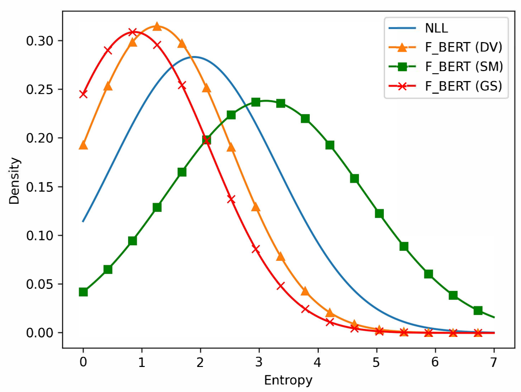

Figure 1: Entropy of the probability vectors generated by the different approaches over the de-en test set.

4.3 Results

Table 1 shows the main results over the respective test sets. As expected, fine-tuning the baseline with the proposed approach has generally helped improve the F_{BERT} scores. However, Table 1 also shows that it has often led to improvements in BLEU score. In the majority of cases, the best results have been obtained with the Gumbel-Softmax (GS), with more marked improvements for de-en and en-tr (+0.36 pp BLEU and +0.76 pp F_{BERT} and +0.72 pp MS for de-en, and +0.58 pp BLEU, +0.20 pp F_{BERT} and +0.03 pp MS for en-tr). Conversely, the dense vectors (DV) and sparsemax (SM) have not been as effective, with the exception of the dense vectors with the zh-en dataset (+0.25 pp BLEU, +0.49 pp F_{BERT} and +0.54 pp MS). This suggests that the Gumbel-Softmax sampling may have played a useful role in exploring alternative word candidates. In fairness, none of the proposed approaches has obtained significant improvements with the en-es dataset. This might be due to the fact that the baseline is much stronger to start with, and thus more difficult to improve upon. In general, both the embedding-based metrics (i.e., F_{BERT} and MS) have ranked the approaches in the same order, with the exception of the en-es dataset.

[2] https://github.com/ijauregiCMCRC/fairseq-bert-loss
[3] https://huggingface.co/models

To provide further insights, similarly to Baziotis et al. (2020), in Figure 1 we plot the distribution of the entropy of the probability vectors generated by the different approaches during inference over the de-en test set. Lower values of entropy correspond to sparser predictions. The plot shows that the models fine-tuned with the dense vectors and the Gumbel-Softmax have made test-time predictions that have been sparser on average than those of the baseline, with the Gumbel-Softmax being the sparsest, as expected. Conversely, and somehow unexpectedly, the model fine-tuned with the sparsemax has made predictions denser than the baseline's. We argue that this may be due to the scale of the logits that might have countered the aimed sparsification of the sparsemax operator. In all cases, the sparsity of the predictions seems to have positively correlated with the improvements in accuracy. For a qualitative analysis, Appendix B presents and discusses various comparative examples for different language pairs.

Finally, Figure 2 shows the effect of the proposed objective over the measured metrics on the de-en validation set at different fine-tuning steps. The plots show that the model rapidly improves the performance in F_{BERT} and MS scores during the first epoch (steps $1 - 967$), peaking in the second epoch ($\approx$ step $1,200$). After that, the performance of the model starts dropping, getting back to the baseline levels in epoch 4. This suggests that training can be limited to a few epochs only, to prevent overfitting. On the other hand, the plots also show a trade-off between the metrics, as the model's improvements in F_{BERT} and MS come at cost of a decrease in BLEU. However, this phenomenon has not been visible on the test set, where all the fine-tuned models have outperformed the baseline also in BLEU score. This suggests that for this dataset the distributions of the training and test sets may be more alike.

5 Conclusion

In this work, we have proposed fine-tuning NMT models with BERTScore, a recently proposed word embedding-based evaluation metric aimed to overcome the typical limitations of n-gram matching. To be able to use BERTScore as an objective function while keeping the model end-to-end differentiable, we have proposed generating *soft* predictions with differentiable operators such as the sparsemax and the Gumbel-Softmax. The experimental results over four language pairs have showed that the proposed approach – nicknamed *BERTTune* – has been able to achieve statistically significant improvements in BLEU, F_{BERT} and MS scores over a strong baseline. As future work, we intend to explore the impact of key factors such as the dataset size, the sparsity degree of the predictions and the choice of different pretrained language models, and we also plan to evaluate the use of beam search/sequential sampling during training to leverage further exploration of candidate translations.

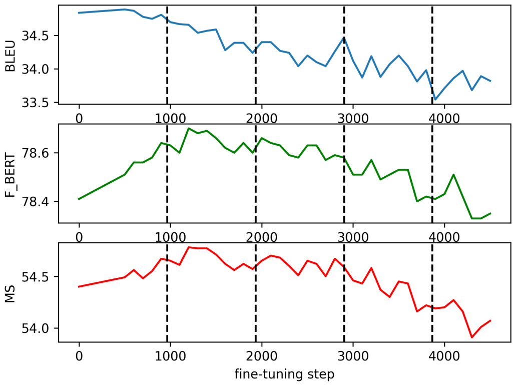

Figure 2: BLEU, F_{BERT} and MS scores of the BERT-Tune (GS) model over the de-en validation set at different fine-tuning steps. Step 0 is the score of the baseline model, and the vertical dashed lines delimit the epochs.

References

Christos Baziotis, Barry Haddow, and Alexandra Birch. 2020. Language model prior for low-resource neural machine translation. In *Proceedings of the Conference on Empirical Methods for Natural Language Processing*.

Katsuki Chousa, Katsuhito Sudoh, and Satoshi Nakamura. 2018. Training neural machine translation using word embedding-based loss. *arXiv preprint arXiv:1807.11219*.

Stéphane Clinchant, Kweon Woo Jung, and Vassilina Nikoulina. 2019. On the use of bert for neural machine translation. In *Proceedings of the 3rd Workshop on Neural Generation and Translation*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Rotem Dror, Gili Baumer, Segev Shlomov, and Roi Reichart. 2018. The hitchhiker's guide to testing statistical significance in natural language processing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, pages 1383–1392.

Sergey Edunov, Alexei Baevski, and Michael Auli. 2019. Pre-trained language model representations for language generation. In *Proceedings of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Sergey Edunov, Myle Ott, Michael Auli, David Grangier, and Marc'Aurelio Ranzato. 2018. Classical structured prediction losses for sequence to sequence learning. In *Proceedings of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Maha Elbayad, Laurent Besacier, and Jakob Verbeek. 2018. Token-level and sequence-level loss smoothing for RNN language models. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, pages 2094–2103.

Eric Jang, Shixiang Gu, and Ben Poole. 2017. Categorical reparameterization with gumbel-softmax. In *Proceedings of the International Conference on Learning Representations*.

Inigo Jauregi Unanue, Ehsan Zare Borzeshi, Nazanin Esmaili, and Massimo Piccardi. 2019. ReWE: Regressing word embeddings for regularization of neural machine translation systems. In *Proceedings of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 430–436.

Diederik P Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*.

Philipp Koehn and Rebecca Knowles. 2017. Six challenges for neural machine translation. In *Proceedings of the First Workshop on Neural Machine Translation*.

Sachin Kumar and Yulia Tsvetkov. 2019. Von mises-fisher loss for training sequence to sequence models with continuous outputs. In *Proceedings of the International Conference on Learning Representations*.

Siyao Li, Deren Lei, Pengda Qin, and William Yang Wang. 2019. Deep reinforcement learning with distributional semantic rewards for abstractive summarization. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Shuming Ma, Xu Sun, Yizhong Wang, and Junyang Lin. 2018. Bag-of-words as target for neural machine translation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, pages 332–338.

Christopher Maddison, Andriy Mnih, and Yee Teh. 2017. The concrete distribution: A continuous relaxation of discrete random variables. In *Proceedings of the International Conference on Learning Representations*.

Andre Martins and Ramon Astudillo. 2016. From softmax to sparsemax: A sparse model of attention and multi-label classification. In *International Conference on Machine Learning*, pages 1614–1623.

Lesly Miculicich, Dhananjay Ram, Nikolaos Pappas, and James Henderson. 2018. Document-level neural machine translation with hierarchical attention networks. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019. fairseq: A fast, extensible toolkit for sequence modeling. In *Proceedings of NAACL-HLT 2019: Demonstrations*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training. *URL https://s3-us-west-2.amazonaws.com/openai-assets/researchcovers/languageunsupervised/language understanding paper. pdf*.

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. 2016. Sequence level training with recurrent neural networks. In *Proceedings of the International Conference on Learning Representations*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 1715–1725.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Proceedings of the Advances in Neural Information Processing Systems*, pages 3104–3112.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proceedings of Advances in Neural Information Processing Systems*, pages 5998–6008.

Sam Wiseman and Alexander M. Rush. 2016. Sequence-to-sequence learning as beam-search optimization. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1296–1306.

Thomas Wolf, Julien Chaumond, Lysandre Debut, Victor Sanh, Clement Delangue, Anthony Moi, Pierric Cistac, Morgan Funtowicz, Joe Davison, Sam Shleifer, et al. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45.

Lijun Wu, Shufang Xie, Yingce Xia, Yang Fan, Jian-Huang Lai, Tao Qin, and Tie-Yan Liu. 2020. Sequence generation with mixed representations. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, pages 10388–10398.

Yingce Xia, Tianyu He, Xu Tan, Fei Tian, Di He, and Tao Qin. 2019. Tied transformers: Neural machine translation with shared encoder and decoder. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 5466–5473.

Jiacheng Yang, Mingxuan Wang, Hao Zhou, Chengqi Zhao, Weinan Zhang, Yong Yu, and Lei Li. 2020. Towards making the most of bert in neural machine translation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 9378–9385.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q Weinberger, and Yoav Artzi. 2020. Bertscore: Evaluating text generation with bert. In *Proceedings of the International Conference on Learning Representations*.

Wei Zhao, Maxime Peyrard, Fei Liu, Yang Gao, Christian M. Meyer, and Steffen Eger. 2019. MoverScore: Text generation evaluating with contextualized embeddings and earth mover distance. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*, pages 563–578.

Jinhua Zhu, Yingce Xia, Lijun Wu, Di He, Tao Qin, Wengang Zhou, Houqiang Li, and Tie-Yan Liu. 2020. Incorporating bert into neural machine translation. In *Proceedings of the International Conference on Learning Representations*.

Appendix A: Preprocessing and hyperparameters

This appendix provides detailed information about the preprocessing of the datasets and the hyperparameter selection to facilitate the reproducibility of the experiments. All the code will be released after the anonymity period.

As part of the preprocessing of the datasets, all sentences have been tokenized and lowercased. The source languages have been tokenized with the Moses tokenizer[4], except for Chinese that has been tokenized using Jieba[5]. The target languages have instead been tokenized with the tokenizer learned by the pretrained language model. As language models for BERTScore, we have used `bert-base-uncased` (en), `dbmdz/bert-base-turkish-uncased` (tr) and `dccuchile/bert-base-spanish-wwm-uncased` (es) from Hugging Face. As language model for the MoverScore, we have used the suggested language model for English[6], `dbmdz/distilbert-base-turkish-cased` for Turkish and `mrm8488/distill-bert-base-spanish -wwm-cased-finetuned-spa-squad2-es` for Spanish, the last two from Huggingface. The few sentences longer than 175 tokens have been removed from all datasets as in the original fairseq preprocessing script. Additionally, further tokenization at subword level has been performed over the source languages using byte-pair encoding (BPE) (Sennrich et al., 2016) with $32,000$ merge operations. An important step in the preprocessing has been to force the decoder and the language model to share the same vocabulary. Therefore, we have assigned the decoder with the vocabulary from the selected pretrained language model, ensuring that both used identical `bos`, `eos`, `pad` and `unk` tokens.

For training a strong transformer baseline, we have followed the recommendations in fairseq[7]. The architecture is the predefined `transformer_iwslt_de_en` architecture (79M parameters) with word embedding and hidden vector dimension size of 512, and 6 transformer layers. We have set the training batch size to $4,096$ tokens, the dropout rate to 0.3 and the `clip_norm` gradient clipping parameter to 0.0. The objective function is the label-smoothed negative log-likelihood, with the smoothing factor set to 0.1. We have used the Adam optimizer (Kingma and Ba, 2015) with a $5e-4$ learning rate and beta values $\beta_1 = 0.9$ and $\beta_2 = 0.98$. We have set the warm-up steps to $4,000$ with an initial learning rate of $1e-7$. During training, we have reduced the learning rate with an inverse square-root scheduler and the weight decay set to 0.001. We have trained the model until convergence of the BLEU score on the validation set, with checkpoints at each epoch and patience set to 3, or until the learning rate dropped below $1e-9$.

For fine-tuning with F_{BERT}, we have initialized the transformer models with the trained weights of the baseline. We have kept all hyperparameters identical, except for the learning rate which has been reduced by an order of magnitude to $5e-5$, following common fine-tuning strategies. The models have been fine-tuned until convergence over the validation set, with patience set to 3. Since the changes have only involved the training objective, the number of trainable parameters has remained exactly the same (79M). At test time, we have used beam search decoding with beam size 5 and length penalty 1.

For all the experiments we have used an NVIDIA Quadro P5000 GPU card with 16 GB of memory.

Appendix B: Translation examples

This appendix shows a few translation examples from the de-en and zh-en language pairs to provide further insights into the behavior of the different models.

The example in Table 2 shows that only the BERTTune model with the Gumbel-Softmax has been able to translate phrases such as *at the moment* and *it was as if / it was like*. This model seems to have been able to capture the exact meaning of the source German sentence, even though it has translated it with a slightly different wording (note that the Gumbel-Softmax fine-tuning explores a larger variety of predictions). The other BERTTune models, too, have translated this sentence better than the baseline.

In the example in Table 3, the baseline has not been able to correctly pick the name of the artist (*bono*, lowercased from *Bono*), choosing instead word *bonobos* (primates). All the BERTTune mod-

[4] https://github.com/moses-smt/mosesdecoder

[5] https://github.com/fxsjy/jieba

[6] https://github.com/AIPHES/emnlp19-moverscore/releases/download/0.6/MNLI_BERT.zip

[7] https://fairseq.readthedocs.io/en/latest/index.html

els have instead made the correct prediction. In this example, it is possible that the BERTTune models have benefited from the fine-tuning with a pretrained language model: word *bono* might not have been present in the limited translation training data, but might have been encountered in the large unsupervised corpora used to train the language model. Another possibility is that they have simply used the copy mechanism more effectively.

In the example in Table 4, all the BERTTune models have correctly translated the phrase *part of the national statistics*, while the baseline has incorrectly translated it as *part of the world record*. In turn, the BERTTune models have translated the phrase *in a decade or two* as *in 10 or 20 years* which is a correct paraphrase, whereas the baseline has used the exact phrase as the reference. We also note that although both the baseline and BERTune translations have scored a BLEU score of 0.0 in this case, the F_{BERT} score has been able to differentiate between them, assigning a score of 72.36 to the BERTTune translation and 72.10 to the baseline. This also shows that small gains in F_{BERT} score can correspond to significant improvements in translation quality.

Finally, in the example in Table 5 only the BERTTune models with dense vectors and Gumbel-Softmax have been able to translate the beginning of the sentence (*i was the guy beaten up*) with acceptable paraphrases (i.e. *and i was the kind of person who had been beaten up / i was that guy who had been beaten*). Conversely, the baseline has translated the ending part of the sentence (*until one teacher saved my life*) with a phrase of antithetical meaning (*until a teacher turned me into this kind of life*).

Src:	in dem moment war es , als ob ein filmregisseur einen bühnenwechsel verlangt hätte .
Ref:	at that moment , it was as if a film director called for a set change .
Transformer NMT:	the moment a film director would have asked a stager .
BERTTune (Dense vectors)	and at that moment , a film director would have wanted a stage change .
BERTTune (Sparsemax)	the moment a film director wanted a stage change .
BERTTune (Gumbel-Softmax)	at the moment , it was like a film director would have wanted a stage change .

Table 2: De-en translation example.

Src:	und interessanterweise ist bono auch ein ted prize gewinner .
Ref:	and interestingly enough , bono is also a ted prize winner .
Transformer NMT:	and interestingly , bonobos are also a ted prize winner .
BERTTune (Dense vectors)	and interestingly , bono is also a ted prize winner .
BERTTune (Sparsemax)	and interestingly , bono is also a ted prize winner .
BERTTune (Gumbel-Softmax)	and interestingly , bono is also a ted prize winner .

Table 3: Another de-en translation example.

Src:	我想在十年或二十年内，这将会成为国家统计数据的一部分。
Ref:	this is going to be , i think , within the next decade or two , part of national statistics .
Transformer NMT:	i think it ’ s going to be part of the world record in a decade or two .
BERTTune (Dense vectors)	and i think that in 10 or 20 years , this will be part of the national statistics .
BERTTune (Sparsemax)	i think that in 10 or 20 years , this will be part of the national statistics .
BERTTune (Gumbel-Softmax)	i think that in 10 or 20 years , this will be part of the national statistics .

Table 4: Zh-en translation example.

Src:	我是那种每周在男生宿舍被打到出血的那种人直到一个老师把我从这种生活中解救出来。
Ref:	i was the guy beaten up bloody every week in the boys ’ room , until one teacher saved my life .
Transformer NMT:	i was the one who was in the dorm room every week , and it wasn ’ t until a teacher turned me into this kind of life .
BERTTune (Dense vectors)	and i was the kind of person who had been beaten up in the dorma every week until a teacher turned me out of this life .
BERTTune (Sparsemax)	i ’ m the kind of person who fell into his dorm room till a teacher turned me through this kind of life .
BERTTune (Gumbel-Softmax)	i was that guy who had been beaten in his dorm room every week, until a teacher took me out of that life .

Table 5: Another zh-en translation example.