Better Neural Machine Translation by Extracting Linguistic Information from BERT

Hassan S. Shavarani **Anoop Sarkar**
School of Computing Science
Simon Fraser University
BC, Canada
{sshavara,anoop}@sfu.ca

Abstract

Adding linguistic information (syntax or semantics) to neural machine translation (NMT) has mostly focused on using point estimates from pre-trained models. Directly using the capacity of massive pre-trained contextual word embedding models such as BERT (Devlin et al., 2019) has been marginally useful in NMT because effective fine-tuning is difficult to obtain for NMT without making training brittle and unreliable. We augment NMT by extracting dense fine-tuned vector-based linguistic information from BERT instead of using point estimates. Experimental results show that our method of incorporating linguistic information helps NMT to generalize better in a variety of training contexts and is no more difficult to train than conventional Transformer-based NMT.

1 Introduction

Probing studies into large contextual word embeddings such as BERT (Devlin et al., 2019) have shown that these deep multi-layer models essentially reconstruct the traditional NLP pipeline capturing syntax and semantics (Jawahar et al., 2019); information such as part-of-speech tags, constituents, dependencies, semantic roles, coreference resolution information (Tenney et al., 2019a,b) and subject-verb agreement information can be reconstructed from BERT embeddings (Goldberg, 2019). In this work, we wish to extract the relevant pieces of linguistic information related to various levels of syntax from BERT in the form of dense vectors and then use these vectors as linguistic "experts" that neural machine translation (NMT) models can consult during translation.

But can syntax help improving NMT? Linzen et al. (2016); Kuncoro et al. (2018); Sundararaman et al. (2019) have reported that learning grammatical structure of sentences can lead to higher levels of performance in NLP models. In particular, Sennrich and Haddow (2016) show that augmenting NMT models with explicit linguistic annotations improves translation quality.

BERT embeddings have been previously considered for improving NMT models. Clinchant et al. (2019) replace the encoder token embedding layer in a Transformer NMT model with BERT contextual embeddings. They also experiment with initializing all the encoder layers of the translation model with BERT parameters, in which case they report results on both freezing and fine-tuning the encoder parameters during training. In their experiments BERT embeddings can help with noisy inputs to the NMT model, but otherwise do not help improving NMT performance.

Imamura and Sumita (2019) suggest that replacing the encoder layer with BERT embeddings and fine-tuning BERT while training the decoder leads to a *catastrophic forgetting* phenomenon where useful information in BERT is lost due to the magnitude and number of updates necessary for training the translation decoder and fine-tuning BERT. They present a two-step optimization regime in which the first step freezes the BERT parameters and trains only the decoder while the next step fine-tunes the encoder (BERT) and the decoder at the same time. Yang et al. (2020) also try to address the *catastrophic forgetting* phenomenon by thinking of BERT as a teacher for the encoder of the neural translation model (student network) (Hinton et al., 2015). They propose a dynamic switching gate implemented as a linear combination of the encoded embeddings from BERT and the encoder of NMT. However these papers do not really focus on the linguistic information in BERT, but rather try to combine pre-trained BERT and NMT encoder representations.

Sundararaman et al. (2019) identify *part-of-speech*, *case*, and *sub-word position* as essential lin-

Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics, pages 2772–2783
April 19 - 23, 2021.

guistic information to improve the quality of both BERT and the neural translation model. They extract each linguistic feature using the Viterbi output of separate models, embed the extracted linguistic information (similar to trained word embeddings) and append these vectors to the token embeddings. However, their model uses point estimates of the syntactic models and they do not use the linguistic information in BERT embeddings.

Weng et al. (2019) use multiple multi-layer perceptron (MLP) modules to combine the information from different layers of BERT into the translation model. To make the most out of the fused information, they also alter the translation model training objective to contain auxiliary knowledge distillation (Hinton et al., 2015) parts concerned with the information coming from the pre-trained language model. Zhu et al. (2020) also inject BERT into all layers of the translation model rather than only input embeddings. Their model uses an attention module to dynamically control how each layer interacts with the representations. In both of these works, the training of the Transformer for NMT becomes quite brittle and is prone to diverge to local optima.

In this paper, we propose using pre-trained BERT as a source of linguistic information rather than a source of frozen pre-trained contextual embedding. We identify components of the BERT embeddings that correspond to different types of linguistic information such as part-of-speech, etc. and fine-tune dense vector embeddings for these linguistic aspects of the input and use them within an NMT model. Our approach does not radically complicate the Transformer NMT model training process both in terms of time and hardware requirements and also in terms of training difficulty (avoids bad local optima).

Our contributions are as follows: (1) A method of linguistic information extraction from BERT which needs supervision while training but works without supervision afterwards. (2) An easily trainable procedure for integrating the extracted information into the translation model. (3) Evaluation of the proposed model on small, medium and large translation datasets.

The source code and trained aspect extractors are available at https://github.com/sfu-natlang/SFUTranslate and our experiments can be replicated using scripts under `resources/exp-scripts/aspect_exps`.

2 NMT and BERT

Machine translation is the problem of transforming an input utterance sequence X in source language l_f into another utterance sequence Y (possibly with varying length) in target language l_e. Machine translation models search among all possible sequences in target language to find the most probable sequence based on the probability distribution of Equation 1.

$$P(y|X, y \in l_e) = \prod_{i=0}^{|max\ len|} p(y_i|X, y_0, ..., y_{i-1}) \quad (1)$$

Neural machine translation (NMT) tries to model the probability distribution $p(y|X)$ using neural networks by taking advantage of deep learning techniques. Transformers (Vaswani et al., 2017) are one type of encoder-decoder neural networks used for translation tasks. In Transformers, the input (in one-hot format) is passed through N layers of encoder and N layers of decoder. In each layer, the layer input passes through multiple attention heads (h heads; each considered a specialist in a different sentence-level linguistic attribute) and then gets transformed to the input for the next layer using a two layer feed-forward perceptron module with input size of d_{model} and hidden layer size of d_{ff}. The final probability distribution $p(y|X)$ is generated using an affine transformation applied to the output of the last feed-forward module in the N^{th} decoder layer. Please see (Vaswani et al., 2017) for further details.

BERT (Devlin et al., 2019) adopts the encoder part of the transformer model and requires training it on large amounts of text data using a *masked language model* objective over sub-words $p(y_i|X, y_0, ..., y_{i-1}, y_{i+1}, ..., y_{max\ len})$ instead of guessing the next sub-word $p(y_i|X, y_0, ..., y_{i-1})$. This bidirectional context turns BERT into a provider of strong contextual sub-word embeddings in many languages. These massively over-parameterized neural networks have revolutionized many different NLP tasks. Effective application of BERT in NMT has been studied in a number of contemporary research projects; Language Modeling, Named Entity Recognition, Question Answering, Natural Language Inference, Text Classification (Devlin et al., 2019), and Question Generation (Chan and Fan, 2019). We approach this problem from the novel perspective of extracting linguistic information encoded in BERT and applying such information in NMT.

3 Linguistic Aspect Extraction from BERT

Since BERT contextual embeddings contain a variety of information (linguistic and non-linguistic), extraction of relevant information plays an important role in further improvement of the downstream tasks. In the rest of this section, we define *aspect vectors* as single-purpose dense vectors of extracted linguistic information from BERT, discuss how aspect vectors can be extracted, and explain how to integrate aspect vectors into NMT.

3.1 Aspect Vectors

To start the information extraction process, we initially need to choose a limited (desired) set of linguistic attributes to look for in BERT embeddings. This attribute set can contain a number of linguistic aspects (e.g. part-of-speech). Each linguistic aspect itself will be defined over a possible aspect tag set (e.g. the set of {*NOUN*, *ADJ*, ...} in part-of-speech). In this paper, we show a linguistic attribute set with $\mathbb{A}$, show a generic aspect with a and point to its relative tag set with t_a.

Given the definition of a linguistic aspect and inspired by the information bottleneck idea (Tishby and Zaslavsky, 2015), we define an *aspect vector* as a single-purpose dense vector extracted from BERT and containing information about a certain linguistic aspect of a particular (sub-word) token in the input sequence. Aspect vectors can be interpreted as feature values equivalent to a specific key (aspect).

3.2 Aspect Vector Extraction

For each embedding vector $\mathbf{E}$ and linguistic aspect a, we define M_a as an aspect-extraction function where $\mathbf{e}_a = M_a(\mathbf{E})$ is a single-purpose dense vector containing maximum aspect information and minimum irrelevant other information.

We ensure the aspect encoding power of $\mathbf{e}_a$ by retrieving its equivalent tag in t_a using a classifier. The aspect prediction loss for a linguistic attribute set $\mathbb{A}$ of size n can be calculated as the average cross entropy loss ($\mathcal{L}_{CE}$) between the classifier prediction and the expected aspect tags for each aspect (Equation 2).

$$\mathcal{L}_a = \frac{1}{n}\sum_{i=0}^{|n|}\mathcal{L}_{CE}^i \tag{2}$$

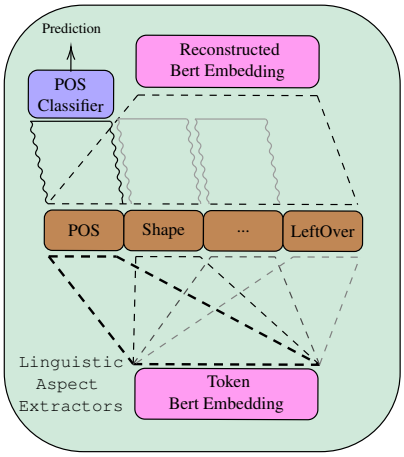

Figure 1: Schematic Aspect Extraction from BERT

We also ensure information integrity[1] of $\mathbf{e}_a$ by concatenating all the aspects (in addition to a "*left-over*" aspect equivalent to all the other non-interesting information) and reconstructing the original embedding vector $\mathbf{E}$ from them[2] in reconstruction vector $\mathbf{R}$. The reconstruction loss ($\mathcal{L}_r$) for the extracted aspect vectors can be calculated as the euclidean distance of the reconstruction vector $\mathbf{R}$ and the original embedding vector $\mathbf{E}$ (Equation 3).

$$\mathcal{L}_r = ||\mathbf{R} - \mathbf{E}||^2 \tag{3}$$

In addition, since our aspect extractor is similar in architecture to a multi-head attention module (with a difference in the fact that we know what exactly each head will be responsible for), to prevent learning redundant representations (Michel et al., 2019), we add the average euclidean similarity ($\mathcal{L}_s$) of each pair of aspect vectors to the training loss function (Equation 4).

$$\mathcal{L}_s = 1 - \left(\frac{1}{n(n-1)}\sum_{i=0}^{|n|}\sum_{j\neq i=0}^{|n|}||e_i - e_j||^2\right) \tag{4}$$

The aspect extractor will be trained over the accumulation of the three mentioned loss components (Equation 5). Figure 1 demonstrates different parts of the aspect extractor and their connections.

$$\mathcal{L}_{fe} = \mathcal{L}_a + \mathcal{L}_r + \mathcal{L}_s \tag{5}$$

[1] We don't expect M_a to change the information inside $\mathbf{E}$ but rather to extract the relevant information.

[2] This idea is analogous to stack-propagation (Zhang and Weiss, 2016) in which propagating the information loss for two tasks helps improving the quality of the encoded representations.

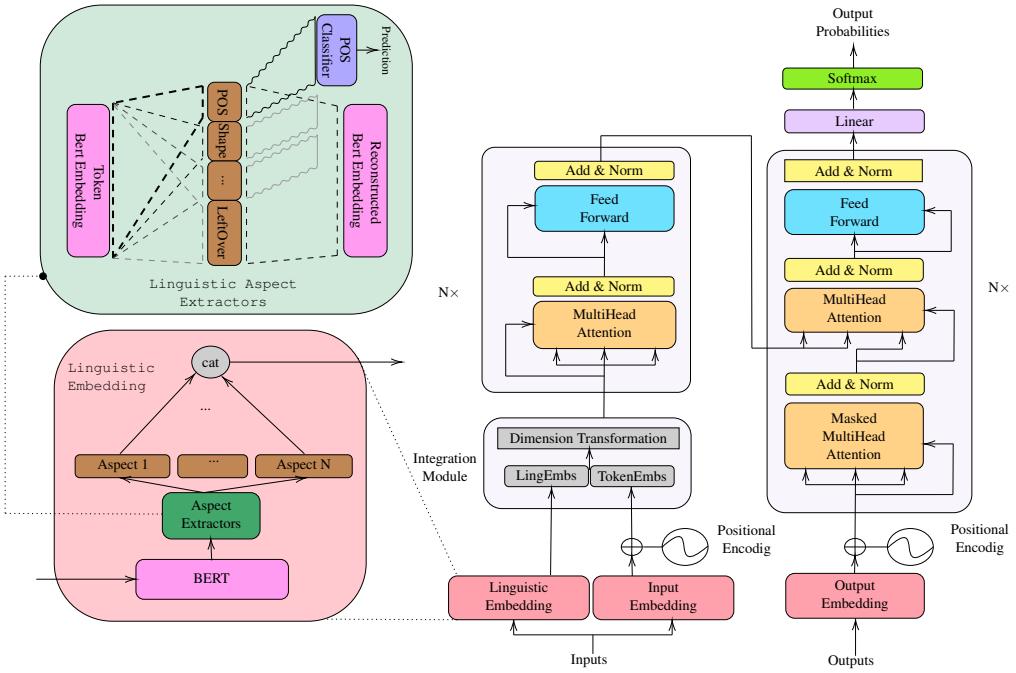

Figure 2: Integration of Extracted Aspect Vectors into NMT. The right hand side part of this figure is taken from Vaswani et al. (2017).

As another important point, a pre-trained BERT model has multiple encoder layers as well as an embedding layer. Choosing the proper layer which contains all of our desired aspects is not simply possible since different layers specialize in different linguistic aspects (Jawahar et al., 2019; Tenney et al., 2019a).

Therefore, as Peters et al. (2018) suggest, we define BERT embedding vector $\mathbf{E}$ as a weighted sum of all BERT layers (of size ℓ) using Equation 6 where α weights are learnable parameters and will be trained along with the other aspect extractor parameters.

$$\mathbf{E} = \sum_{j=0}^{\ell} \alpha_j \mathbf{E}_j^{BERT} \tag{6}$$

3.3 Integrating Aspect Vectors into NMT

Once the aspect vectors are created, we throw away the classifiers and the reconstruction layers and place the encoder part of our trained aspect extractor (the mapping from BERT contextual embeddings to aspect vectors) in an input integration module designed to augment the neural translation model input with aspect vectors[3].

The integration module (constructed using a two layer perceptron network) receives the concatenated aspect vectors (we call this concatenated vector a linguistic embedding[4]) and the token embedding (inherited from the Transformer model), and maps the linguistic embedding into a vector of the same size as the token embedding. Then, it projects the concatenation of both embeddings to a vector with the same size as the token embedding of the original Transformer model[5]. Figure 2 demonstrates this process.

4 Experiments

In this section, we initially examine our designed aspect extractor and report its classification accuracy scores. Next, we integrate the extracted aspect vectors into the neural machine translation framework as explained in Section 3.3 and study the effects of integrated vectors on the performance of the models.

4.1 Data

We choose three German (which has explicit and nuanced linguistic features) to English datasets in different data sizes to examine our proposed framework.

[3]We use the same sub-word model in pre-trained BERT to provide sub-word tokens to our NMT model.

[4]This embedding vector can be similar to what a factor token contains in Factored-NMT (García-Martínez et al., 2016) with a difference that it is generated in the space of linguistic aspects and does not need an embedding layer.

[5]This step is necessary to prevent any change in other parts of the model which would make comparison of the results unfair due to effects on the number of parameters and the learning capability of the model.

We use Multi30k (M30k)[6] as our small dataset. This dataset contains a multilingual set of image descriptions in German, English and French. Due to this reason, we also consider experimenting on German to French as our second small dataset. The M30k data contains 29K training sentences, 1014 validation sentences (val) and 1000 test sentences ($test2016$).

We take IWSLT (Cettolo et al., 2012)[7] as our medium sized dataset. The sentences in this dataset are quite different from M30k since they are composed from the transcriptions of TED talks as well as dialogues and lectures[8]. The IWSLT data contains 208K training sentences, 888 validation sentences ($dev2010$) and multiple test sets ($tst2010$ to $tst2015$ with 1568, 1433, 1700, 993, 1305, and 1080 sentences, respectively).

For the large data size, we consider WMT[9], a large (4.5M training sentences) set of parallel sentences from the proceedings of the European Parliament as well as web crawled news articles. We remove 0.05% of the training data (2290 sentences; lines with numbers divisible by 2000) and use it as the validation set (we call it wmt_val) and take $newstest$ data from 2014 to 2019 as our test sets (with 3003, 2169, 2999, 3004, 2998 and 1997 sentences, respectively).

We remove train data sentences longer than 100 words and uncase and normalize both side sentences using `MosesPunctNormalizer`[10] before tokenization. The reference side of the test data remains untouched in all the steps of our experiments.

4.2 Linguistic Aspect Vector Extraction

In this section, we study our linguistic aspect extractor training procedure and analyze the quality of the extracted aspect vectors.

We choose our linguistic attribute set ($\mathbb{A}$) as Sundararaman et al. (2019) suggest, however, we replace '*case*' with '*word-shape*'[11] since we believe the complete shape of the word is much more informative specially in sub-word settings. In addition, we consider a two-level hierarchy in part-of-speech tags to benefit from both higher accuracy in exploring the syntactic search space and lower model confusion in cases where the fine-grained tags are not helpful. Therefore, we consider coarse-grained and fine-grained *part-of-speech* (CPOS and FPOS), *word-shape* (WSH), and *sub-word position*[12] (SWP) to form our experimental linguistic attribute set ($\mathbb{A}$). Other linguistic attributes such as dependency parses or sentiment could be considered as aspects in our model but we leave that for future work.

We use the spaCy German tagger[13] model to acquire our intended linguistic aspect labels. Since spaCy is trained on word-level while BERT is trained on sub-word level, we had to align the sequences using a monotonic alignment algorithm (see Appendix A.1.1). The fine-grained part-of-speech tagger in spaCy[14] is pre-trained on TIGER Corpus[15] (Smith et al., 2003) and inherits its 55 fine-grained tags from TIGER treebank. The coarse-grained spaCy part-of-speech tagger has been trained by defining a direct mapping from 55 tags of the TIGER treebank to the 16 tags in the Universal Dependencies v2 POS tag set[16].

We use a 12-layer[17] German pre-trained BERT model for encoding the source sentences in aspect extractors. We use an uncased model as our translation model performs on lowercased data and the results are recased using the moses recaser so that the results are cased BLEU scores comparable to other systems[18]. We pass the BERT-encoded source sentences through a single perceptron middle layer of size 1000. We divide the output of this layer to

[6] AKA *Flickr30K* provided in task 1 of WMT17 multimodal machine translation, http://www.statmt.org/wmt17/multimodal-task.html

[7] 2017 was the last year that the data for this task got updated; https://wit3.fbk.eu/mt.php?release=2017-01-mted-test

[8] While the talks are quite polished, they still contain many verbal structures and sometimes even sounds (e.g. "`Imagine an engine going clack, clack, clack, clack, clack, clack, clack.`").

[9] Europarl+CommonCrawl+NewsCommentary https://www.statmt.org/wmt14/translation-task.html, please note that in the later years this training set remained the same, but ParaCrawl data was added to it. We do not use ParaCrawl data since it is quite noisy and we aim to limit the effects of uncontrolled variables in our training data. However, we report our results on all the test tests after 2014.

[10] https://github.com/alvations/sacremoses/

[11] Representing capitalization (changing alphabet to `x` or `X`), punctuation, and digits (changing digits to `d`). As an example for *word-shape*, the sub-word *##arxiv.* in the token '*myarxiv.org*' will turn to *##xxxxx.*.

[12] Encoding the word with one of the three labels "`Begin`", "`Inside`", or "`Single`".

[13] https://spacy.io/models/de

[14] SpaCy reports 96.52% accuracy for this model.

[15] https://www.ims.uni-stuttgart.de/

[16] https://universaldependencies.org/v2/postags.html

[17] Hidden state size of 768 with 12 heads; written in PyTorch and distributed by Wolf et al. (2019). You can find model configurations in https://github.com/dbmdz/berts.

[18] We recommend using a cased BERT model for translation systems that handle casing differently.

	Sub-word Level					Word Level			
	CPOS	**FPOS**	**WSH**	**SWP**	**#tokens**	**CPOS**	**FPOS**	**WSH**	**#tokens**
M30k	96.88	96.18	99.79	99.93	16096	97.95	97.34	99.74	12823
IWSLT	92.69	90.48	99.73	97.14	22687	94.84	93.07	99.69	19039
WMT	92.64	91.60	97.74	98.94	70139	94.86	94.01	97.38	55135

Table 1: F-1 scores acquired after training the aspect extractor on German side of parallel data and passing the validation sets of each data set through trained aspect extractors. The `#tokens` column shows the number of tokens in the validation set.

Aspect Extractor Training Data	**FPOS**	**SWP**
M30k	79.39	90.63
IWSLT	77.80	88.34
WMT	82.13	91.42
TIGER	84.64	92.64

Table 2: F-1 scores of fine-grained part-of-speech prediction of TIGER corpus test data (BERT encoded) fed to each of the trained aspect classifiers. The scores are calculated over a total of 7516 sub-word tokens in 358 test sentences of TIGER. Extractors trained on M30k, IWSLT, and WMT have not been provided with any part of TIGER before evaluation.

'*number of aspects + 1*' splits to form our desired aspect vectors (of size 200). Please see Appendix A.1.1 for more implementation details.

We train three different aspect extractors, one for each dataset and feed in the source sentences of the dataset to our model in batches of size 32 for 3 epochs[19]. Table 1 shows F-1 scores of classifying the validation set data using different aspect vectors after training the aspect extractors on the train set sentences. Please note that for calculating the word-level scores, in cases of disagreement between different sub-word tokens, the sub-word prediction of the first sub-word token has been counted as the prediction for the word label.

We also validate our trained (on M30k, IWSLT, and WMT) aspect extractors against the manual annotations of TIGER treebank with which the spaCy fine-grained part-of-speech tagger has been trained. We train an extra aspect extractor using the train set of TIGER corpus and test all four trained aspect extractors against TIGER data test set[20]. This experiment evaluates the absolute power of our simple feed-forward aspect extractors in performing the aspect classification task. Please note that our goal in this experiment is not to achieve the state-of-the-art fine-grained part-of-speech tagging results as our aspect extractors receive their input from BERT and do not directly access the tagged input sentences. Table 2 contains the results of comparison between predictions of different aspect extractor classifiers and TIGER gold labels.

4.3 Uniqueness of Information in Linguistic Aspect Vectors

Considering the high F-scores for each aspect category in each dataset (Table 1), we can conclude that our aspect extractor maximizes the relevant information extraction from BERT embeddings. The loss in Equation 4 maximizes the distance between aspect vectors. To test whether this leads to a diverse set of aspect vectors, each specialized to their own linguistic attributes, we consider each aspect category a, after training the aspect extractors. We take each of the other extracted aspect vectors a' (except the "*left-over*" vector) and use each of them to train a new classifier[21] that predicts the right class for category a based on aspect vector a'. This will test the correlation between the information in aspect vectors a' and the tags in category a. If the classification scores for this counterfactual test are high then our model has failed in fine-tuning each aspect vector to predict a particular linguistic aspect. We compare the classification scores to a trivial baseline: predict the most frequent class always. Table 3 shows the results of this counterfactual test on the aspect extractor trained on TIGER data. We can see that the average F-1 scores are very low when we use counterfactual aspect vectors to predict a linguistic aspect on which it was not fine-tuned (e.g. use aspect vector trained on part-

[19]Since the number of WMT sentences are much bigger, we stop training WMT aspect extractors when there is no improvement in aspect classification result (rounded to have 3 decimal places) of any label for at least 40 batches.

[20]We use `german_tiger_test_gs.conll` in the version of TIGER released in *2006 CoNLL Shared Task - Ten Languages*. Both train and test data are accessible through https://catalog.ldc.upenn.edu/LDC2015T11.

[21]We thank the anonymous reviewers for their valuable feedback on this procedure.

TIGER test	Sub-word Level			
	CPOS	FPOS	WSH	SWP
most frequent class	NOUN	NN	xxxx	single
percentage in total	27.12	27.07	39.07	59.92
average classification F-1	1.89	0.23	12.20	42.97
#tokens	$7516 \times 3 = 22548$			

Table 3: Classification scores of each aspect classifier when fed with other extracted aspect vectors. We expect the F-1 scores to be low so we can conclude that our aspect extractor truly excludes irrelevant information from each aspect.

of-speech to predict word shape). This shows that our training method fine-tunes each aspect vector to its linguistic task.

To validate the loss in Equation 3, we calculate the average euclidean distance of the aspect extractor reconstructed vectors and the original BERT embedding vectors[22] for M30k German to English dataset. We unit normalize each of the vectors for a score in $[0, 1]$. The average euclidean distance value of 0.1863 tells us that the reconstruction component of the aspect extractor is capable of reconstructing vectors that are close to the original embedding vectors.

4.4 Linguistic Aspect Integrated Machine Translation

After confirming the adequacy and uniqueness of linguistic information in aspect vectors, we integrate the encoder part of aspect extractors into the translation model and perform translation experiments on M30k, IWSLT, and WMT datasets. In our experiments, we compare our model to three baselines : (1) the vanilla transformer model (Vaswani et al., 2017) which does not use any external source of information, (2) the syntax-infused transformer model (Sundararaman et al., 2019) which explicitly embeds linguistic aspect labels and concatenates their embedding to the token embedding, (3) the transformer model with bert-freeze input setting (Clinchant et al., 2019) which replaces the input embedding layer of the encoder module in transformer with a fully pre-trained BERT model. Appendix A.1.2 provides the configurations and sufficient details for replication of our experiments in this section.

During each training trial, we perform 9 validation set evaluation steps (one after visiting each 10% of the data). In each step, the validation set is translated with the current state of the model (at the time of evaluation) and the generated sentences are detokenized and compared to the validation set reference data to produce sentence-level BLEU (Lin and Och, 2004) scores. The best scoring model throughout training is selected as the model with which the test set(s) are translated.

For M30k and IWSLT data sets, we train two separate models, one using the aspect vectors trained on the source side of its own training data (in-domain) and the other using the aspect vectors trained on the source side of WMT data (out-of-domain). We use cased BLEU (evaluated with the standard `mteval-v14.pl` script) and METEOR (Denkowski and Lavie, 2014) to compare different models. Tables 4 and 7 show the results of evaluating the models trained with different mentioned settings.

The evaluation results show that taking advantage of aspect vectors improves the accuracy of translating German to both English and French in M30k as well as German to English in IWSLT and WMT. Also, in majority of the cases WMT-trained aspect vectors have pushed the model to produce more accurate results since they contain more generalized information. Based on these results, we conjecture that aspect vectors trained on large out-of-domain data can be helpful in low-resource settings but we leave the examination of this idea for future work.

Aside from performance, our model is approximately 5 times faster than syntax-infused translation model (Sundararaman et al., 2019) while demanding less number of trainable parameters. Although it is not as fast as bert-freeze model (Clinchant et al., 2019) in large settings (because of the size of computations required for calculating the linguistic embedding), it is comparable in speed to bert-freeze in medium and small scale settings. Appendix A.2 contains some additional insights regarding how aspect vectors can help translation systems trained on different dataset sizes.

Tables 5 and 6 demonstrate some examples of cases where aspect vectors has been useful in improving the translation quality.

[22] Average results of Equation 3 for all the tokens in the train set.

a) M30k†	German to English				German to French			
	val	test2016	#param	runtime*	val	test2016	#param	runtime*
Vaswani et al. 2017	39.63	38.35	9.5 M	84 min	31.07	30.29	9.4 M	93 min
Sundararaman et al. 2019	40.03	38.32	13.9 M	514 min	32.55	32.71	13.6 M	504 min
Clinchant et al. 2019 (bert freeze)	40.07	39.73	9.1 M	99 min	33.83	33.15	9.0 M	104 min
Aspect Augmented +M30k asp. vectors	**40.47**	40.19	10.1 M	104 min	34.45	**34.42**	9.9 M	108 min
Aspect Augmented +WMT asp. vectors	38.72	**41.53**	10.1 M	102 min	**34.73**	34.28	9.9 M	118 min

b) IWSLT†	dev2010	tst2010	tst2011	tst2012	tst2013	tst2014	tst2015	#param	runtime*
Vaswani et al. 2017	27.69	27.93	31.88	28.15	29.59	25.66	26.76	18.4 M	172 min
Sundararaman et al. 2019	29.53	29.67	33.11	29.42	30.89	27.09	27.78	28.9 M	1418 min
Clinchant et al. 2019 (bert freeze)	30.31	30.00	34.20	30.04	31.26	27.50	27.88	18.0 M	212 min
Aspect Augmented +IWSLT asp. vectors	29.03	29.17	33.42	29.58	30.63	26.86	27.83	18.9 M	214 min
Aspect Augmented +WMT asp. vectors	**31.22**	**30.82**	**34.79**	**30.29**	**32.34**	**27.71**	**28.40**	18.9 M	211 min

c) WMT†	wmt_val	nt2014	nt2015	nt2016	nt2017	nt2018	nt2019	#param	runtime*
Vaswani et al. 2017	28.96	26.91	26.93	31.42	28.07	33.56	29.77	68.7 M	35 h
Sundararaman et al. 2019	28.56	27.80	26.93	30.44	28.63	33.87	30.48	93.8 M	258 h
Clinchant et al. 2019 (bert freeze)	28.63	27.54	27.15	31.69	28.30	33.89	**31.48**	69.1 M	33 h
Aspect Augmented +WMT asp. vectors	**28.98**	**28.05**	**27.58**	**32.29**	**29.07**	**34.74**	**31.48**	70.3 M	46 h

Table 4: Evaluated cased BLEU score (calculated using `mteval-v14.pl` script) results on M30k, IWSLT, and WMT datasets. `#param` represents the number of trainable parameters (size of BERT model parameters [110.5M] has not been added to the model size for the aspect augmented and bert-freeze models since BERT is not trained in these settings). `runtime` is the total time the training script has ran and includes time taken for reading the data and training the model from scratch (iterating over the instances for all the epochs).
All the baseline results are achieved using our re-implementation of the mentioned papers.
* We have used a single GeForce GTX 1080 GPU for M30k experiments and a single Titan RTX GPU for IWSLT and WMT experiments.
† Each experiment was repeated three times, and we report the average in this table.

5 Conclusion and Future Work

In this paper, we proposed a simple method of extracting linguistic information from BERT contextual embeddings and integrating them into neural machine translation framework. We showed that the linguistic aspect vectors provide the translation models with out-of-domain knowledge which not only improves the translation quality but also helps the model to better deal with out-of-vocabulary words. In the future, we would like to reconsider the integration module as a multi-head attention module, except that it will attend to different linguistic aspects of the current sub-word or sub-word tokens of a single word. Increasing the number of linguistic aspects (especially the use of syntactic dependencies and morphology) and studying the effects of the aspect vector size on the quality of generated translations are other directions of future research. We would also like to examine the effectiveness of aspect vectors trained on large out-of-domain data in low-resource settings and explore the effects of using linguistic aspect vectors in tasks other than machine translation.

Acknowledgments

We would like to thank the anonymous reviewers for their helpful comments. The research was partially supported by the Natural Sciences and Engineering Research Council of Canada grants NSERC RGPIN-2018-06437 and RGPAS-2018-522574 and a Department of National Defence (DND) and NSERC grant DGDND-2018-00025.

Source	Ihm **werde weiterhin vorgeworfen**, unerlaubt geheime Informationen weitergegeben zu haben.
Reference	He **is still accused of passing on** secret information without authorisation.
Vaswani et al. 2017	He has **also been accused of having** illegally **passed on** secret information.
Clinchant et al. 2019	He **continues to be accused of** fraudulently **passing on** secret information.
Sundararaman et al. 2019	He **is also accused of having pass** unauthorised secret information **on**.
Aspect Augmented NMT	He **is still accused of passing on** illegal secret information.
Source	Auto und Traktor krachen zusammen: Frau stirbt bei schrecklichem Unfall
Reference	Car and tractor **crash together**: woman **dies** in terrible accident
Vaswani et al. 2017	Car and traktor **cranes together**: women **die** in the event of a terrible accident.
Clinchant et al. 2019	Cars and tractors **are killing** women in the event of a terrible accident.
Sundararaman et al. 2019	Auto and tractor **are blowing together**: woman **dies** when the terrible accident occurs.
Aspect Augmented NMT	Car and tractor **crash together**: woman **dies** in terrible accidents.

Table 5: Examples of improved translation quality of WMT data where *part-of-speech* aspect vectors have helped the model choose better words both syntactically and semantically.

Source	Bucht die besten Hostels in **Ouarzazate** über Hostelsclub.
Reference	Book the best hostels in **Ouarzazate** with Hostelsclub.
Vaswani et al. 2017	Book the best hostels in **ouarzazazate** with Hostelsclub.
Clinchant et al. 2019	Book the best hostels in **Ouarzate** with Hostelsclub.
Sundararaman et al. 2019	Book the best hostels in **ouarzazazate** with Hostelsclub.
Aspect Augmented NMT	Book the best hostels in **Ouarzazate** with Hostelsclub.
Source	Die **Deutsche Bahn** will im kommenden Jahr die Kinzigtal-Bahnstrecke verbessern.
Reference	The **Deutsche Bahn** hopes to improve the Kinzigtal railway line in the coming year.
Vaswani et al. 2017	**The German Railway** wants to improve the Kinzig valley railway line next year.
Clinchant et al. 2019	**Christian Deutsche Bahn** intends to improve the Kinzig valley railway next year.
Sundararaman et al. 2019	**The German Railway** wants to improve the kinziggia railway line next year.
Aspect Augmented NMT	**Deutsche Bahn** wants to improve the Kinzig valley railway in the coming year.

Table 6: Examples of improved translation quality of WMT data where *word-shape* and *sub-word position* aspect vectors have helped the model choose a better sequence of sub-words when it faces out-of-vocabulary tokens.

a) M30k†	**German to English**		**German to French**	
	val	**test2016**	**val**	**test2016**
Vaswani et al. 2017	37.20	36.56	53.22	52.58
Sundararaman et al. 2019	38.14	37.13	54.18	54.37
Clinchant et al. 2019 (bert freeze)	38.44	37.42	55.10	54.50
Aspect Augmented +M30k asp. vectors	**39.22**	38.17	**56.21**	**56.40**
Aspect Augmented +WMT asp. vectors	38.90	**38.57**	56.12	55.98

b) IWSLT†	**dev2010**	**tst2010**	**tst2011**	**tst2012**	**tst2013**	**tst2014**	**tst2015**
Vaswani et al. 2017	31.82	31.99	34.57	32.65	32.49	30.65	31.13
Sundararaman et al. 2019	32.91	32.95	35.35	33.10	33.17	31.32	31.90
Clinchant et al. 2019 (bert freeze)	33.34	32.78	35.42	33.12	33.20	31.22	31.45
Aspect Augmented +IWSLT asp. vectors	32.86	32.86	35.38	33.43	33.23	31.37	31.87
Aspect Augmented +WMT asp. vectors	**33.78**	**33.56**	**36.14**	**33.51**	**33.98**	**31.86**	**32.37**

c) WMT†	**wmt_val**	**nt2014**	**nt2015**	**nt2016**	**nt2017**	**nt2018**	**nt2019**
Vaswani et al. 2017	**30.65**	33.80	33.70	**37.10**	34.44	37.81	36.05
Sundararaman et al. 2019	29.23	31.57	31.61	34.05	31.87	35.18	33.60
Clinchant et al. 2019 (bert freeze)	30.39	33.46	33.20	36.13	33.73	37.24	35.68
Aspect Augmented +WMT asp. vectors	30.61	**33.97**	**33.99**	37.01	**34.71**	**38.17**	**36.48**

Table 7: Evaluated METEOR score (calculated using the tool provided by Alon Lavie (https://www.cs.cmu.edu/~alavie/METEOR/; version 1.5)) results on M30k, IWSLT, and WMT datasets.
† Each experiment was repeated three times, and we report the average in this table.

References

Mauro Cettolo, Christian Girardi, and Marcello Federico. 2012. Wit^3: Web inventory of transcribed and translated talks. In *Proceedings of the 16th Conference of the European Association for Machine Translation (EAMT)*, pages 261–268, Trento, Italy.

Ying-Hong Chan and Yao-Chung Fan. 2019. BERT for question generation. In *Proceedings of the 12th International Conference on Natural Language Generation*, pages 173–177, Tokyo, Japan. Association for Computational Linguistics.

Stephane 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*, pages 108–117, Hong Kong. Association for Computational Linguistics.

Michael Denkowski and Alon Lavie. 2014. Meteor universal: Language specific translation evaluation for any target language. In *Proceedings of the EACL 2014 Workshop on Statistical Machine 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 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

William A. Gale and Kenneth W. Church. 1993. A program for aligning sentences in bilingual corpora. *Computational Linguistics*, 19(1):75–102.

Mercedes García-Martínez, Loïc Barrault, and Fethi Bougares. 2016. Factored neural machine translation architectures. In *HAL archives ouvertes*.

Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256.

Yoav Goldberg. 2019. Assessing bert's syntactic abilities. *Computation and Language Research Repository*, arXiv:1901.05287. Version 1.

Geoffrey Hinton, Oriol Vinyals, and Jeffrey Dean. 2015. Distilling the knowledge in a neural network. In *NIPS Deep Learning and Representation Learning Workshop*.

Kenji Imamura and Eiichiro Sumita. 2019. Recycling a pre-trained BERT encoder for neural machine translation. In *Proceedings of the 3rd Workshop on Neural Generation and Translation*, pages 23–31, Hong Kong. Association for Computational Linguistics.

Ganesh Jawahar, Benoît Sagot, and Djamé Seddah. 2019. What does BERT learn about the structure of language? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3651–3657, Florence, Italy. Association for Computational Linguistics.

Adhiguna Kuncoro, Chris Dyer, John Hale, Dani Yogatama, Stephen Clark, and Phil Blunsom. 2018. LSTMs can learn syntax-sensitive dependencies well, but modeling structure makes them better. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1426–1436, Melbourne, Australia. Association for Computational Linguistics.

Chin-Yew Lin and Franz Josef Och. 2004. ORANGE: a method for evaluating automatic evaluation metrics for machine translation. In *COLING 2004: Proceedings of the 20th International Conference on Computational Linguistics*, pages 501–507, Geneva, Switzerland. COLING.

Tal Linzen, Emmanuel Dupoux, and Yoav Goldberg. 2016. Assessing the ability of LSTMs to learn syntax-sensitive dependencies. *Transactions of the Association for Computational Linguistics*, 4:521–535.

Paul Michel, Omer Levy, and Graham Neubig. 2019. Are sixteen heads really better than one? In *Advances in Neural Information Processing Systems*, pages 14014–14024.

Myle Ott, Sergey Edunov, David Grangier, and Michael Auli. 2018. Scaling neural machine translation. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 1–9, Brussels, Belgium. Association for Computational Linguistics.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana. Association for Computational Linguistics.

Ofir Press and Lior Wolf. 2017. Using the output embedding to improve language models. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 157–163, Valencia, Spain. Association for Computational Linguistics.

Alexander Rush. 2018. The annotated transformer. In *Proceedings of Workshop for NLP Open Source Software (NLP-OSS)*, pages 52–60, Melbourne, Australia. Association for Computational Linguistics.

Mike Schuster and Kaisuke Nakajima. 2012. Japanese and korean voice search. In *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5149–5152. IEEE.

Rico Sennrich and Barry Haddow. 2016. Linguistic input features improve neural machine translation. In *Proceedings of the First Conference on Machine Translation: Volume 1, Research Papers*, pages 83–91, Berlin, Germany. Association for Computational Linguistics.

George Smith et al. 2003. A brief introduction to the tiger treebank, version 1. *Potsdam Universität*.

Dhanasekar Sundararaman, Vivek Subramanian, Guoyin Wang, Shijing Si, Dinghan Shen, Dong Wang, and Lawrence Carin. 2019. Syntax-infused transformer and bert models for machine translation and natural language understanding. *Computation and Language Research Repository*, arXiv:1911.06156. Version 1.

Ian Tenney, Dipanjan Das, and Ellie Pavlick. 2019a. BERT rediscovers the classical NLP pipeline. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4593–4601, Florence, Italy. Association for Computational Linguistics.

Ian Tenney, Patrick Xia, Berlin Chen, Alex Wang, Adam Poliak, R Thomas McCoy, Najoung Kim, Benjamin Van Durme, Sam Bowman, Dipanjan Das, and Ellie Pavlick. 2019b. What do you learn from context? probing for sentence structure in contextualized word representations. In *International Conference on Learning Representations*.

Naftali Tishby and Noga Zaslavsky. 2015. Deep learning and the information bottleneck principle. In *2015 IEEE Information Theory Workshop (ITW)*, pages 1–5. IEEE.

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 *Advances in neural information processing systems*, pages 5998–6008.

Rongxiang Weng, Heng Yu, Shujian Huang, Shanbo Cheng, and Weihua Luo. 2019. Acquiring knowledge from pre-trained model to neural machine translation. *AAAI*.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, R'emi Louf, Morgan Funtowicz, and Jamie Brew. 2019. Huggingface's transformers: State-of-the-art natural language processing. *Computation and Language Research Repository*, arXiv:1910.03771. Version 1.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *Computation and Language Research Repository*, arXiv:1609.08144. Version 2.

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. *AAAI*.

Yuan Zhang and David Weiss. 2016. Stack-propagation: Improved representation learning for syntax. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1557–1566, Berlin, Germany. Association for Computational Linguistics.

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. *ICLR 2020*.

A Appendices

A.1 Implementation Details

In this section, we provide implementation details that could not be placed in the main write-up due to space limitations, but we believe are quite helpful for replication of our work. We divide this section into two parts, one focused on linguistic aspect vector extraction (Section 4.2) and the other on linguistic aspect integrated machine translation (Section 4.4).

A.1.1 Linguistic Aspect Vector Extraction Implementation Details

The pre-trained spaCy tagger that we used in our experiments is trained on the word-level while the pre-trained BERT operates on sub-word level[23]. The two sequences need to be aligned, so we can assign aspect attributes to BERT sub-word tokens. Inspired by Gale and Church (1993), we align the two sequences using a heuristic divide-and-conquer monotonic alignment technique which finds the parts of the two sequences that are certainly equal and aligns the parts in between using recursive calls to itself[24].

Next, we explain how we implement the aspect extractors. We implement our aspect extractors using PyTorch framework and initialize them using Xavier initialization (Glorot and Bengio, 2010). We perform backpropagation using SGD (initial learning rate of 0.05, momentum value of 0.9, gradient clip norm of 5.0). To cope with inequality in the frequency of the different tags in each aspect tag set (t_a, see §3.1), we practice weighted

[23]The alignment is non-trivial e.g. "`hadn't`" is tokenized to "`hadn`" and "`'t`" by spaCy and to "`had`" and "`n't`" by BERT, causing many-to-many alignments.

[24]https://github.com/sfu-natlang/SFUTranslate/translate/readers/sequence_alignment.py

Dataset	WMT	IWSLT	M30k
N	6	6	4
d_{model}	512	256	256
d_{ff}	2048	512	512
h	8	4	4
opt factor	1	2	1
opt warmup	4000	8000	2000
grad accumulation	8	2	1
batch size*	4096	4096	2560
epochs	7	20	20

Table 8: The transformer model settings for each dataset given the training data size. "N" is the number of layers in both encoder and decoder. Please see §2 for more information about model parameters.
*The maximum number of sub-word tokens per batch.

backpropagation with weights proportional to the inverse frequency of each tag. We decay learning rate with a factor of 0.9 when the loss value stops improving.

A.1.2 Linguistic Aspect Integrated Machine Translation Implementation Details

We implement our baseline transformer model using the guidelines suggested by Rush (2018) in our translation toolkit SFUTranslate and extend it for implementing the aspect-augmented model as well as the syntax-infused transformer and transformer with bert-freeze input setting. Table 8 provides the configuration settings for each of the models used in our experiments.

We use the pre-trained WordPiece[25] (Schuster and Nakajima, 2012) tokenizer packaged and shipped with BERT (containing 31,102 sub-word tokens for German language) to tokenize the source side data, and tokenize the target side data with MosesTokenizer[26] followed by the same WordPiece tokenizer model, trained on target data, to split the target tokens into sub-tokens. We set the target side WordPiece vocabulary size to 30,000 sub-words for English and French. Our models share the vocabulary and embedding modules of both source and target (Press and Wolf, 2017) since both source and target are trained in sub-word space. The shared vocabulary sizes of M30k (German to English), M30k (German to French), IWSLT, and WMT are 16645, 16074, 40807, 47940, respectively.

We generate target sentences using beam search with beam size 4 and length normalization factor (Wu et al., 2016) of 0.6. We merge the WordPiece tokens in the generated sentences (a post-processing step to create words) and use MosesDetokenizer[27] to detokenize the generated outputs. We use Moses recaser[28] to produce cased translation outputs. We use `mteval-v14.pl` script for cased BLEU evaluation.

For all models, we set positional encoding max length to 4096, dropout to 0.1, loss prediction smoothing to 0.1, and initialize the models using Xavier initialization (Glorot and Bengio, 2010). We train all models using `NoamOpt` optimizer (Rush, 2018) and perform the gradient accumulation trick (Ott et al., 2018) with one update per a number of batches (Table 8; `grad accumulation`) to simulate larger batch sizes on a single GPU.

A.2 Additional Analysis of Linguistic Aspect Integrated Machine Translation Results

In this section, we analyze the results of our aspect integrated translation experiments. We provide our analysis in two parts, one for small and medium sized datasets and the other for large ones.

For smaller datasets (containing a few hundred thousand sentence pairs or less), the broader perspective of BERT knowledge is helpful in limiting the search space for the model. So using our technique, the translation model receives more information regarding the general use cases of (locally) rare words. Linguistic aspect vectors also help the model better understand less familiar (in comparison to what is frequent in its limited size training data) syntactic structures in input sentences. This is why we believe aspect vectors can be helpful in low-resource settings.

Improving models with large amounts of data (with several million sentence pairs) is a challenging task. The best practice in training neural translation models is to initialize the embedding module with small random values and let the model search through the parameter space to find the optimal parameter settings. Extracted aspect vectors, as an external source of monolingual knowledge on the source side, are a more reasonable starting point for large models than random initialization. Integrating aspect vectors thus helps these models find a better path towards the optimal point(s) and increases the chances of the model ending up in a more desirable point in search space.

[25] https://github.com/huggingface/tokenizers
[26] https://github.com/alvations/sacremoses
[27] https://github.com/alvations/sacremoses
[28] https://github.com/moses-smt/mosesdecoder