From bilingual to multilingual neural-based machine translation by incremental training

Carlos Escolano | Marta R. Costa-Jussà | José A. R. Fonollosa

TALP Research Center, Universitat Politècnica de Catalunya, Barcelona, Spain

Correspondence
Marta R. Costa-Jussà, TALP Research Center, Universitat Politècnica de Catalunya, Barcelona, Spain. Email: marta.ruiz@upc.edu

Funding information
Agencia Estatal de Investigación/ Spanish Ministerio de Economía y Competitividad: Ramón y Cajal, Grant/Award Numbers: PCIN-2017-079, EUR2019-103819, PID2019-107579RB-I00; European Regional Development Fund: Ramón y CajalGoogle Faculty Research Award

Abstract
A common intermediate language representation in neural machine translation can be used to extend bilingual systems by incremental training. We propose a new architecture based on introducing an interlingual loss as an additional training objective. By adding and forcing this interlingual loss, we can train multiple encoders and decoders for each language, sharing among them a common intermediate representation. Translation results on the low-resource tasks (Turkish-English and Kazakh-English tasks) show a BLEU improvement of up to 2.8 points. However, results on a larger dataset (Russian-English and Kazakh-English) show BLEU losses of a similar amount. While our system provides improvements only for the low-resource tasks in terms of translation quality, our system is capable of quickly deploying new language pairs without the need to retrain the rest of the system, which may be a game changer in some situations. Specifically, what is most relevant regarding our architecture is that it is capable of: reducing the number of production systems, with respect to the number of languages, from quadratic to linear; incrementally adding a new language to the system without retraining the languages already there; and allowing for translations from the new language to all the others present in the system.

1 | INTRODUCTION

Machine translation—in a highly multilingual environment—poses several challenges, such as the quadratic growth of the number of possible combinations of translation directions. Among these challenges are the acquisition and curation of parallel data and the allocation of hardware resources for training and inference purposes. This situation becomes even worse given that the translation quality depends strongly on the amount of available training data when offering translation for a language pair for which there is little or no parallel data available.

Neural machine translation (NMT) (Cho et al., 2014; Sutskever, Vinyals, and Le, 2014) has arisen as a completely new paradigm for machine translation outperforming previous statistical approaches (Koehn, Och, and Marcu, 2003) in most of the tasks. One clear exception is low-resource tasks (Koehn and Knowles, 2017), where statistical machine translation can still outperform or be competitive with NMT (Artetxe, Labaka, and Agirre, 2018; Lample, Ott, Conneau, Denoyer, and Ranzato, 2018).

Among others, one clear advantage of NMT is that it opens up new challenges in machine translation, such as multimodal machine translation (Elliott, Frank, Barrault, Bougares, and Specia, 2017) or the finding of a common intermediate representation that allows training single encoders and decoders for each language.
Another novelty of our architecture is that, in the optimization process, we are add a loss term. This loss term is the correlation between intermediate representations from different languages. In this way, we force the system to learn the intermediate representation while training multiple translation systems. One of the challenges at this point is to find a suitable intermediate representation distance function. To address this, we propose and evaluate different distance measures. The results on the Workshop on Machine Translation (WMT) benchmark for low-resource tasks (English, Turkish and Kazakh) show that our architecture (with the new loss term based on the correlation distance and without variational autoencoders) produces competitive translations and achieves improvements over the baseline system while benefiting from the fact that it can be easily and inexpensively extended to new languages. Regarding the addition of new languages, we show that our architecture is capable of scaling to new source languages without requiring the retraining of all languages in the system.

Contributions: This paper proposes a proof of concept of a new multilingual NMT approach. The current approach is based on joint training without parameter sharing by enforcing a compatible representation between the jointly trained languages and using multitask learning. We propose effectively using the correlation as the distance between intermediate representations. This approach is shown to offer a strategy scalable to new languages without needing to retrain any of the previous languages in the system and enabling zero-shot translation. Our architecture while being much more flexible and allowing for quickly deployment of new languages is also able to improve the BLEU results over these of the baseline systems.

Organization: The rest of the paper is organized as follows. Section 2 reports the most closely related works on the topic. Next, Section 3 presents the necessary background to make the manuscript self-contained. Section 4 details the architecture proposed in this study, both the joint training and how to scale to new languages. Section 5 describes the data and implementation used in the experiments, and it reports the translation results. Section 6 provides an analysis of how the system is able to recover a sentence from the intermediate representation and offers an insightful visualization of the intermediate representation. Finally, Section 7 draws the most relevant conclusions of this study.

2 RELATED WORK

In this work, we focus on training a common language representation with deep learning techniques. The objective is to train an intermediate representation that allows using independent encoders and decoders for each
language. In this scenario, translation systems in a highly multilingual environment get reduced from quadratic to linear and also, translation is available for language pairs that have not been explicitly trained. Following a similar objective or methodology, the related works include the following ones.

**Shared encoders/decoders:** Johnson et al. (2016) feed a single encoder and decoder with multiple input and output languages. With this approach, the authors show that zero-shot learning is possible. The authors show by means of visualizing similar sentences in different languages that there is some evidence indicating the somewhat close appearance of these sentences in the common representation. More recently, Arivazhagan et al. (2019) propose auxiliary losses on the NMT encoder that impose representational invariance across languages, which is shown to improve zero-shot translation.

**Dedicated encoder/decoder:** These approaches have varying architectures, for example, from many encoders to one decoder (many-to-one) (Zoph and Knight, 2016), from one encoder to many decoders (one-to-many) (Dong, Wu, He, Yu, and Wang, 2015) and from one encoder to one decoder (one-to-one), the latter of which we focus on here because these approaches are closest to ours. Firat, Cho, Sankaran, Vural, and Bengio (2017) propose extending the classical recurrent NMT bilingual architecture (Bahdanau, Cho, and Bengio, 2015) to a multilingual architecture by designing a single encoder and decoder for each language with a shared attention-based mechanism. Schwenk and Douze (2017) and España-Bonet, Varga, Barrón-Cedeño, and van Genabith (2017) evaluate how a recurrent NMT architecture without attention is able to generate a common representation between languages. These authors use the inner product or cosine distance to evaluate the distance between sentence representations. Recently, Lu et al. (2018) suggest training single encoders and decoders for each language to generate interlingual embeddings that are agnostic related to the input and output languages.

**Other related architectures:** While unsupervised machine translation (Artetxe et al., 2017; Lample et al., 2017) does not directly pursue a common intermediate representation, but it is somewhat related to our approach. Artetxe et al. (2017) and Lample et al. (2017) propose a translation system that is able to operate while trained only on a monolingual corpus. The architecture is basically a shared encoder with pretrained embeddings and two decoders (one of which includes an autoencoder). Our work is also related to recent works on sentence representations (Conneau et al., 2017, 2018; Eriguchi, Johnson, Firat, Kazawa, and Macherey, 2018) and on taking advantage of multitask learning (Dong et al., 2015). However, the main difference is that these works aim at extending representations to other natural language processing tasks, while we are aiming at finding the most suitable representation to make interlingual machine translation feasible. While interesting for further research, it is out of the scope study to evaluate and adapt of this intermediate representation to multiple tasks.

### 3 | BACKGROUND

In this section, we report the techniques that are used for the development of the proposed architecture in this paper: variational autoencoders (Kingma and Welling, 2013; Rumelhart et al., 1985; Zhang, Xiong, Su, Duan, and Zhang, 2016) present a different approach in which the objective is to learn the parameters of a probability distribution that characterizes the intermediate representation. This learning allows us to sample new synthetic instances from the distribution and generate them using the decoder part of the network.

#### 3.1 | Variational autoencoders

Autoencoders consist of a generative model that is able to generate its own input. This is useful in training an intermediate representation, which can later be employed as a feature for another task or even as a dimensionality reduction technique, as is the case of traditional autoencoders, which learn to produce an intermediate representation for an existing example. Variational autoencoders (Kingma and Welling, 2013; Rumelhart et al., 1985; Zhang, Xiong, Su, Duan, and Zhang, 2016) present a different approach in which the objective is to learn the parameters of a probability distribution that characterizes the intermediate representation. This learning allows us to sample new synthetic instances from the distribution and generate them using the decoder part of the network.

#### 3.2 | Decomposed vector quantization

One of the strategies to create variational autoencoders is vector quantization (van den Oord et al., 2017). This technique consists in the addition of a table, between the encoder and decoder, of dimension $K \times D$ where $K$ is the number of possible representations and $D$ the dimension or set of dimensions of each of the representations. Given an input representation produced by the encoder, the closest vector from the vector quantization table is retrieved, and fed to the decoder to perform the reconstructions. This table can be considered as the distribution from which all possible input latent representations are sampled, transforming the representation from contiguous to discrete, as only $K$ different vectors can be retrieved from the table.

While the distance between the table and a given encoding is differentiable, assessing the minimum one is not. To train the system end-to-end, gradients are copied...
from the decoder to the encoder instead of being propagated through the vector quantization. As the vector quantization is the closest one to the encoding, these gradients act as an approximation of the actual error and allow the system to be trained end-to-end. Equation (1) shows its loss function, where the first term, \( L_R \), is the reconstruction loss term of the loss function and responsible to update both encoder and decoder and where gradient copying is applied. The second term computes the distance between the encoding \( h(x) \) and the vector quantization \( vq(x) \), to which stop gradients \( sg() \) is applied. The third term is a commitment term to regularize the convergence speed of the encoder and vector quantization table.

\[
L = L_R + \| (h(x) - sg(vq(x))) \|^2_2 + \beta \| (sg(h(x)) - vq(x)) \|^2_2.
\] (1)

As mentioned in Kaiser et al. (2018), this approach may be prone to posterior collapse, where all representations cluster to a small region in space, and just a few values can be retrieved from the table. To resolve this issue, they proposed dvq, which uses a set of \( n \) tables in which each table is used to represent a portion of the representation that is later concatenated and fed to the decoder. This architecture also allows producing more representations than a single table, as they are created as the combination of outputs while maintaining the number of parameters.

### 3.3 Transformer

The current state-of-the-art architecture for NMT is the transformer (Vaswani et al., 2017), which uses multiple self-attention and feed-forward layers to address sequences by inputting the whole sequence at once and using self-attention to attend to the relevant parts of the sequence and solve the coreference issues. Apart from improving the results of the previous sequence-to-sequence systems (Vaswani et al., 2017), transformers can run in parallel. Among other tricks and particularities, this architecture requires summing the positional embeddings, as previously proposed in Gehring, Auli, Grangier, and Dauphin (2017), to word embeddings to explicitly encode the relative positions of tokens, as having a non-recurrent architecture, it does not have an inherent notion of sequentiality.

### 4 Model Architecture

In this section, we report the details of our proposed architecture. We describe the joint training and how we are scaling to new languages.

#### 4.1 Definitions

Before explaining our proposed model, we introduce the annotations that will be used hereinafter. Languages will be denoted by capital letters \( X, Y, \) and \( Z \), while sentences will be denoted by lower-case letters \( x, y, \) and \( z \) given that \( x \in X, y \in Y, \) and \( z \in Z \). Then, sentence \( i \) in the corpus data will be referred to as \( x_i, y_i, \) and \( z_i \).

We will consider as an encoder \( (e_x, e_y, e_z) \) the layers of the network that given an input sentence produce a sentence representation in a space. Analogously, a decoder \( (d_x, d_y, d_z) \) is the layers of the network that given the sentence representation of the source sentence, are able to produce the tokens of the target sentence. Encoders and decoders will be always considered as independent modules that can be arranged and combined individually as no parameter is shared between them. Each language and module has its own weights independent from all the others present in the system.

#### 4.2 Joint training

Given two languages \( X \) and \( Y \), our objective is to train independent encoders and decoders for each language, that is, \( e_x, d_x \) and \( e_y, d_y \) that produce compatible sentence representations. For instance, given a sentence \( x \) in language \( X \), we can obtain a representation \( r_x \) from the encoder \( e_x \) that can be used to either generate a sentence reconstruction using decoder \( d_x \) or a translation using decoder \( d_y \). With this objective in mind, we propose a training schedule that combines two tasks (auto-encoding and translation) and the two translation directions simultaneously by optimizing the following loss:

\[
L = L_{XX} + L_{YY} + L_{XY} + L_{YX} + d
\] (2)

where \( L_{XX} \) and \( L_{YY} \) correspond to the reconstruction losses of both languages \( X \) and \( Y \), respectively (defined as the cross-entropy of the generated tokens and the source sentence for each language); \( L_{XY} \) and \( L_{YX} \) correspond to the translation terms of the loss measuring the token generation of each decoder given a representation generated by the other language decoder (using the cross-entropy between the generated tokens and the translation reference); and \( d \) corresponds to the distance metric between the representation computed by the encoders. This last term forces the representations to be similar without sharing parameters while providing a measure of similarity between the generated spaces. We tested different distance metrics such as \( \text{L}_1, \text{L}_2 \), or the discriminator addition (which tries to predict from which language the representation was generated). For all these alternatives, we experienced a spatial collapse in which all sentences
tend to be located in the same spatial region. This closeness between the sentences of the same language makes them non-informative for decoding. As a consequence, the decoder performs as a language model, producing an output based only on the information provided by the previously decoded tokens. To prevent this collapse, we propose a less restrictive measure based on correlation distance (Chandar, 2015), as computed in Equations (3) and (4). The rationale behind this loss is maximizing the correlation between the representations produced by each language while not enforcing the distance over the individual values of the representations.

\[
d = 1 - c(h(X), h(Y))
\]

\[
c(h(X), h(Y)) = \frac{\sum_{i=1}^{n} (h(x_i) - \bar{h}(X)) \cdot (h(y_i) - \bar{h}(Y))}{\sqrt{\sum_{i} (h(x_i) - \bar{h}(X))^2} \cdot \sqrt{\sum_{i} (h(y_i) - \bar{h}(Y))^2}}
\]

where \( X \) and \( Y \) correspond to the data sources we are trying to represent; \( h(x_i) \) and \( h(y_i) \) correspond to the intermediate representations learned by the network for a given \( i \) observation; and \( \bar{h}(X) \) and \( \bar{h}(Y) \) are, for a given batch, the intermediate representation mean of \( X \) and \( Y \), respectively.

Figure 1 shows the different tasks and directions that the system has been trained to perform and the different directions of the system. Each decoder is able to process the representation produced by each encoder to either translate or reconstruct the source language sentence.

4.3 Joint training with dvq

Measuring the distance can be a difficult task due to the high dimensionality of the transformer hidden states \( h(x) \in \mathbb{R}^{h \cdot t} \) and the fact that in order to compute the vanilla attention at the decoder, we need to keep the contextual representation of each of the \( t \) source tokens.

To facilitate this computation, we propose the addition of dvq (Kaiser et al., 2018) as shown in Figure 2. It consists of splitting the intermediate representations produced into \( n \) parts of equal size \( h(x)_j \in \mathbb{R}^{h/t} \), as done to apply multihead attention. For each split part \( h(x)_j \), we compute the closest vector from the \( j \)th vector quantization embedding table \( vq_j \). The final representation fed into the decoder is the concatenation of all \( vq_j \) vectors, for all sentences in the batch.

Previous work (Kaiser et al., 2018) showed that splitting the embedding table helps the system to be less prone to posterior collapse, where all intermediate representations are in a small neighborhood in space and just a small number of entries from the vector quantization embedding table are retrieved. Additionally, as the tables are independent, the system can produce a higher number of representations by combining different segments compared to the single table approach, while the number of parameters remains constant.

As our objective is to force the intermediate representations from both languages to be as similar as possible, our dvq tables are shared between both languages, to ensure that representations in a similar range from both languages are equally represented after the vector quantization step.

To train our model end-to-end, we have to ensure that gradients are able to flow from the decoder to the encoder despite the distance to the vector quantization table being based on computing the minimum distance to the intermediate representation, which is not differentiable. Following the process proposed in van den Oord et al. (2017), we copy the gradients directly from the decoder to the encoder instead of propagating them from the dvq tables. Because the objective of the representation is to become similar to fixed representations from the tables, even though the gradients are not directly computed over the representation, they are close enough to update the parameters of our
encoders. Equations (5) and (6) show our proposed loss that builds on top of our basic objective from Equation (2), where there is a term for each translation/reconstruction direction, plus a distance measure. Instead of computing each term as a cross-entropy loss, we propose a modification of Equation (1), where we compute its cross-entropy loss and for each encoding $h(x_{ij})$ representing the $i$th token in these sentence, we compute the distance and commitment terms of each encoding $h(x_{ij})$ to the closest vector from the $j$th vector quantization table $vq_j(x_{ij})$.

$$L = \text{LDVQ}_{XX} + \text{LDVQ}_{YY} + \text{LDVQ}_{XY} + \text{LDVQ}_{YX} + d$$  \hspace{1cm} (5)$$

$$\text{LDVQ}_{XX} = L_{XX} + \sum_{i=1}^{l} \sum_{j=1}^{n} \left\| \left( h(x_{ij}) - \text{sgn}(vq_j(x_{ij})) \right) \right\|_2^2 + \beta \left\| \left( \text{sgn}(h(x_{ij})) - vq_j(x_{ij}) \right) \right\|_2^2.$$  \hspace{1cm} (6)

4.4 | Scaling to new languages

Given the model between languages $X$ and $Y$, the following step involves adding new languages in order to use our architecture as a multilingual system. Since parameters are not shared between the independent encoders and decoders, our architecture enables to add new languages without the need to retrain the languages currently in the system. Let us say that we have $Z - X$ parallel data. Then, we can set up a new bilingual system with language $Z$ as source and language $X$ as the target. To ensure that the representation produced by this new pair is compatible with the previously jointly trained system, we use the previous $X$ decoder ($d_x$) as the decoder of the new $ZX$ system and we freeze it. During training, we optimize the cross-entropy between the generated tokens and the language $X$ reference data and update only the layers belonging to the language $Z$ encoder ($e_z$). In doing so, we train $e_z$ not only to produce good quality translations but also representations similar to the already trained languages.

Our training schedule enforces the generation of a compatible representation, which means that the newly trained encoder $e_z$ can be used as input of the decoder $d_y$ from the jointly trained system to produce zero-shot $Z$ to $Y$ translations; see Figure 3 for an illustration. The fact that the system enables zero-shot translation shows that the representations produced by our training schedule contain useful information and that this can be preserved and shared with new languages just by enforcing the new modules to train with the previous one, without any modification to the architecture.

A current limitation is the need to use the same vocabulary for the shared language ($X$) in both training steps. The use of subwords (Sennrich, Haddow, and Birch, 2015) mitigates the impact of this constraint.

5 | EXPERIMENTAL FRAMEWORK

In this section, we provide details on the data and implementation used in the experiments. Additionally, we
report the translation results. The results are presented in terms of the BLEU (Papineni, Roukos, Ward, and Zhu, 2002) which is the standard automatic measure used in machine translation.

5.1 | Data

For the experiments, we used the Turkish-English parallel data from setimes2 (Tiedemann, 2009) which is used in WMT 2017 and the Kazakh-English parallel data from the news domain which is used in WMT 2019. The training set for the Turkish-English data included approximately 200,000 parallel sentences and for the Kazakh-English data, it included approximately 100,000 parallel sentences. As development and test sets, we used newsdev2016 and newstest2016, respectively, for the Turkish-English data and newsdev2019 was split into development and test sets for the Kazakh-English experiments. Additionally, we extracted the Kazakh-Turkish test set from the OPUS database (Tiedemann, 2012) to evaluate the zero-shot translation. For experiments with larger datasets, we used the data shared between the Russian-English case used in WMT 2019 and that between the Russian-Kazakh case. The validation and test sets from the Russian-English case were extracted from the Yandex corpus. The validation set for the Russian-Kazakh case was extracted from news-commentary-v14. Finally, and only for visualization and analysis purposes, we extracted 381 sentences that are multi-way parallel in Turkish-Kazakh-English. These sentences were also extracted from the OPUS database For the latter, we downloaded the Turkish-English and the Kazakh-English datasets and matched the English sentences that were identical. Detailed statistics of the corpus are shown in Table 1.

Preprocessing consisted of a pipeline of punctuation normalization, tokenization, corpus filtering of sentences longer than 80 words and true-casing. These steps were performed using the scripts available from the Moses tools (Koehn et al., 2007). In the experiments with subwords, the preprocessed data were tokenized using byte pair encoding (BPE) (Sennrich, Haddow, and Birch, 2016).

5.2 | Implementation

We used the transformer implementation provided by Fairseq. The parameters varied depending on the configuration of the experiment. For experiments within configuration 1 (conf1), we used six blocks of multihead attention with eight heads each, an embedding/hidden dimensionality of 128 and a fixed learning rate of 0.001 and a vocabulary size of 12,000 words. For experiments involving dvq, 4 tables of 5,000 elements, 32 dimensions each and 1.0 as \( \beta \) for the commitment term of the loss function.

For experiments within configuration 2 (conf2), we used six attention blocks with four heads each, an embedding/hidden dimensionality of 512, and a fixed learning rate of 0.001 and a vocabulary size of 16,000 BPE tokens. For all cases, we used (Kingma and Ba, 2014) as the optimizer. The joint training was performed on two Nvidia Titan X GPUs with 12 GB of RAM, while for the addition of languages, Titan X GPU was used. The systems were trained until non-improvement was seen on the validation set.

5.3 | Translation quality for the joint training

Table 2 shows the BLEU results for the two experimental configurations (1 & 2) as reported in above and for each translation direction from English-to-Turkish (EN-TK)
and from Turkish-to-English (TK-EN). Within the first configuration, we show the results for variations on the proposed architecture (JointTrain) which include both non-variational and variational (dvq) outcomes with the same hyperparameters of the baseline system and a comparison of two distance losses.

**Variational (dvq) versus non-variational:** Using dvq worsens the performance in both directions, and the loss is higher in the EN-TK case. When contrasting the impact of the dvq in our proposed architecture, we see that the performance of nonvariational architecture is higher than that of dvq using any type of distance. However, the loss, when using the correlation distance, is higher in the TK-EN direction than in the opposite direction. Although we report negative results within the integration of variational autoencoders, we believe that it is worth showing that such advanced techniques are specifically not beneficial for the purpose of learning an intermediate representation. While reporting these results, we can show to the reader that this approach is not useful in our multilingual setting.

**Correlation versus maximum distance loss:** Regarding the distance loss, the correlation distance clearly provides better translation results, by approximately 1.5 BLEU in both directions when using the nonvariational architecture. The improvement in the correlation distance compared to that in the maximum distance is even higher when using the variational architecture.

<table>
<thead>
<tr>
<th>Language pair</th>
<th>Corpus</th>
<th>Language Set</th>
<th>Segments</th>
<th>Words (byte pair encoding)</th>
</tr>
</thead>
<tbody>
<tr>
<td>TK-EN</td>
<td>SETIMES</td>
<td>TK Training</td>
<td>200 × 10^3</td>
<td>5,513 × 10^3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>EN Training</td>
<td></td>
<td>5,467 × 10^3</td>
</tr>
<tr>
<td></td>
<td>Newsdev 2017</td>
<td>TK Validation</td>
<td>1,001 × 10^3</td>
<td>26, 7 × 10^3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>EN Validation</td>
<td></td>
<td>28, 4 × 10^3</td>
</tr>
<tr>
<td></td>
<td>Newsdev 2017</td>
<td>TK Test</td>
<td>3 × 10^3</td>
<td>86, 6 × 10^3</td>
</tr>
<tr>
<td>KK-EN</td>
<td>News commentary v14</td>
<td>KK Training</td>
<td>100 × 10^3</td>
<td>2,219 × 10^3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>EN Training</td>
<td></td>
<td>2,216 × 10^3</td>
</tr>
<tr>
<td></td>
<td>Newsdev 2019</td>
<td>KK Training</td>
<td>1,033 × 10^3</td>
<td>36, 7 × 10^3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>EN Validation</td>
<td></td>
<td>36, 7 × 10^3</td>
</tr>
<tr>
<td></td>
<td>Newsdev 2019</td>
<td>KK Test</td>
<td>1,033 × 10^3</td>
<td>38 × 10^3</td>
</tr>
<tr>
<td>KK-TK</td>
<td>OpenSubtitles</td>
<td>KK Test</td>
<td>2, 594 × 10^3</td>
<td>22, 5 × 10^3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TK Test</td>
<td></td>
<td>26, 3 × 10^3</td>
</tr>
<tr>
<td>KK-TK-EN</td>
<td>Tatoeba</td>
<td>KK Test</td>
<td>381</td>
<td>1, 435 × 10^3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>TK</td>
<td></td>
<td>1, 369 × 10^3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>EN</td>
<td></td>
<td>1, 827 × 10^3</td>
</tr>
<tr>
<td>RU-EN</td>
<td>Yandex corpus + ParaCrawl</td>
<td>RU Training</td>
<td>6, 066 × 10^3</td>
<td>180 × 10^3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>EN Training</td>
<td></td>
<td>311 × 10^6</td>
</tr>
<tr>
<td></td>
<td>Newsdev 2019</td>
<td>RU Validation</td>
<td>4 × 10^3</td>
<td>117, 3 × 10^3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>EN Validation</td>
<td></td>
<td>31, 8 × 10^3</td>
</tr>
<tr>
<td></td>
<td>Newsdev 2019</td>
<td>RU Test</td>
<td>1 × 10^3</td>
<td>29, 8 × 10^3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>EN</td>
<td></td>
<td>51, 2 × 10^3</td>
</tr>
<tr>
<td>KK-RU</td>
<td>WMT19 crawled corpus</td>
<td>KK Training</td>
<td>4, 18 × 10^6</td>
<td>284, 9 × 10^6</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RU Training</td>
<td></td>
<td>109, 1 × 10^6</td>
</tr>
<tr>
<td></td>
<td>Ext. WMT19 crawled corpus</td>
<td>KK Validation</td>
<td>4 × 10^3</td>
<td>266, 6 × 10^3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RU Validation</td>
<td></td>
<td>24, 2 × 10^3</td>
</tr>
<tr>
<td></td>
<td>Ext. WMT19 crawled corpus</td>
<td>KK Test</td>
<td>1 × 10^3</td>
<td>66, 9 × 10^3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RU</td>
<td></td>
<td>25, 4 × 10^3</td>
</tr>
</tbody>
</table>
Within this best configuration, our best proposed architecture, which is the nonvariational autoencoder with the correlation distance (JointTrain + corr) shows similar performances to those of the baseline system (transformer).

Up to point, the experiments have been performed using only words. Here, we switch from employing sub-word-NMT (Sennrich et al., 2016) which is the standard tokenization of words. Our configuration follows the standard setup with 16,000 operations and a shared vocabulary between both languages. The second part of Table 2 shows the performance of our best architecture from configuration 1 against that of the baseline system with the second experimental configuration (using BPE as the tokenization and a larger word embedding of 512).

We achieved gains of +0.5 BLEU from EN-TK and +0.7 BLEU from TK-EN over the corresponding baseline. At this point, we tried training our architecture without autoencoders, and we saw that doing so improved our design over the baseline system but not over our complete original proposed joint training. Therefore, training with autoencoders helps to improve translation performance.

Note that the performances of both the baseline and our architecture in this second configuration are higher than the best system results from WMT 2017 (García-Martínez et al., 2017). We compare with the case of using parallel data only, without adding back-translated monolingual data (which were 10.9 for EN-TK and 14.2 for TK-EN).

We can hypothesize that with the second configuration which is larger than the first one, our model is better suited to learn translation and the intermediate representation that we are forcing. This is not possible when using smaller hyperparameters.

### 5.4 Adding new languages and zero-shot translation

At this point, we use the best configuration from the above experiments (configuration 2, using BPE and 512 word embedding dimensions). We add Kazakh as a new language to this system as proposed in Section 4.4. Table 3 shows that the Kazakh-English case performs +0.6 BLEU points over the baseline. The frozen English decoder previously trained using the Turkish-English parallel data may be responsible for the increase in performance.

Finally, another relevant aspect of the proposed architecture is enabling zero-shot translation. To evaluate it, we compare the performance of Kazakh-Turkish compared to that of a pivot system based on the cascade. This system consists of translating from Kazakh to English and from English to Turkish. We can carry out this pivot either with the baseline system (standard transformer) or with our JointTrain architecture modules. Finally, we can use the zero-shot strategy which consists in using a Kazakh encoder and a Turkish decoder. The results show that the zero-shot translation provides slightly lower quality than that of the pivot systems, but the pivot joint training yields an improvement over the pivot baseline.

The results from Table 3 may seem low, but note that our system in the Kazakh-to-English case yields results comparable to those of the systems that participated in the WMT 2019 evaluation, which did not use monolingual data and backtranslate it7.

### 5.5 Larger datasets

One of the main advantages of shared architecture multilingual NMT systems (Johnson et al., 2016) is that they allow

---

**Table 2** BLEU results for the different experimental configurations (1 & 2) and alternative architectures: The transformer system (baseline) and different configurations of our architecture, our joint training (JointTrain) with and without decomposed vector quantization (dvq), and the correlation distance (corr) and the maximum of difference (max)

<table>
<thead>
<tr>
<th>Config</th>
<th>EN-TK</th>
<th>TK-EN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline</td>
<td>8.32</td>
<td>12.03</td>
</tr>
<tr>
<td>Baseline dvq</td>
<td>2.89</td>
<td>8.14</td>
</tr>
<tr>
<td>JointTrain + corr</td>
<td>8.11</td>
<td>12.00</td>
</tr>
<tr>
<td>JointTrain + max</td>
<td>6.19</td>
<td>10.38</td>
</tr>
<tr>
<td>JointTrain + dvq + corr</td>
<td>7.45</td>
<td>7.56</td>
</tr>
<tr>
<td>JointTrain + dvq + max</td>
<td>2.40</td>
<td>5.24</td>
</tr>
<tr>
<td>Baseline</td>
<td>11.85</td>
<td>14.31</td>
</tr>
<tr>
<td>JointTrain+corr</td>
<td>12.56</td>
<td>14.82</td>
</tr>
<tr>
<td>JointTrain+corr+no-autoencoders</td>
<td>12.16</td>
<td>14.21</td>
</tr>
</tbody>
</table>

Note: The best results are in bold.
by parameter sharing the flow of information between high-resource languages and low-resource languages. As a result, the performance of the latter is improved.

In this experiment, we want to confirm that our proposed architecture also presents this property, by adding a new resource language with a frozen model trained on a high-resource language. To do so, we trained a Russian-English model and added Kazakh as a low-resource language.

We added Kazakh to the already trained Russian decoder from the Russian-English trained system.

As the objective of this experiment is only to measure the impact of the additional data for low resources the vocabularies were shared between all three languages, including two different script systems, Latin for English and Cyrillic for Russian and Kazakh. Table 4 shows the results for the baselines and our architecture between Russian and English.

Because the system benefits from a larger amount of available data, we want to evaluate its performance in the case of training Kazakh with a larger dataset and perform zero-shot translation from Kazakh to English. Thus, as mentioned, we add Kazakh to the system by training a Kazakh encoder with the already trained Russian decoder.

The results in Table 5 show that this zero-shot approach outperforms both the baseline and direct Kazakh-to-English addition, proving that having access to more data provides better models in this architecture. This idea holds even when to parallel data are used only during the training period and the translation is based only on the compatibility of the sentence representations.

Note that both Tables 4 and 5 show that our proposed system has a lower performance than that of the baseline system when trained on this new larger dataset. However, our results are quite competitive and we want to emphasize that our architecture has the flexibility of adding new languages to the system without retraining. In some situations, we may be inclined to lose approximately 15% of the performance to achieve fast deployment (e.g., crisis, urgent product deliver).

### 6 | Encoder/Decoder Compatibility and Intermediate Representation Visualization

Our main objective is to create an intermediate representation that can be understood by the different modules trained in the system, where the modules are the encoders

<table>
<thead>
<tr>
<th>Table 3</th>
<th>New supervised language (KK-EN) comparing: The baseline architecture to our added language (AddLang)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Config</td>
<td>System</td>
</tr>
<tr>
<td>2</td>
<td>Baseline</td>
</tr>
<tr>
<td>2</td>
<td>AddLang +corr</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*Note: Zero-shot translation (KK-TK) is provided by our architecture compared to a baseline, which is a pivotal system from KK-EN and EN-TK. The best results are in bold.*

<table>
<thead>
<tr>
<th>Table 4</th>
<th>BLEU results for the larger RU-EN dataset</th>
</tr>
</thead>
<tbody>
<tr>
<td>Config</td>
<td>System</td>
</tr>
<tr>
<td>Baseline</td>
<td>2</td>
</tr>
<tr>
<td>JointTrain+corr</td>
<td>2</td>
</tr>
</tbody>
</table>

*Note: Best results are in bold.*

<table>
<thead>
<tr>
<th>Table 5</th>
<th>Zero-shot translation (KK-EN) provided by our architecture compared to: a low-resource baseline (KK-EN), a pivotal system from KK-RU and from RU-EN, under both the baseline and our JointTrain architecture</th>
</tr>
</thead>
<tbody>
<tr>
<td>Config</td>
<td>System</td>
</tr>
<tr>
<td>2</td>
<td>Baseline</td>
</tr>
<tr>
<td>2</td>
<td>Pivot baseline</td>
</tr>
<tr>
<td>2</td>
<td>Pivot JointTrain</td>
</tr>
<tr>
<td>2</td>
<td>ZeroShot</td>
</tr>
</tbody>
</table>

*Note: Best results are in bold.*
and decoders of all languages involved in the training. However, similar representations may not lead to compatible encoders/decoders. Additionally, different trainings can produce representations with different mean distances in the representations that can generate similar translation outputs. With this focus in mind, in this section, we perform further analysis of our architecture by looking at the compatibility between encoders and decoders and visualizing the intermediate representation. The model used for the analysis within this section is the JointTrain + corr model and configuration 2, which is the best performing model from Table 2 for the Turkish-English and Kazakh-English cases from Table 3.

### 6.1 Encoder/decoder compatibility

We propose the following analysis to measure the compatibility of our encoders and decoders. Given a parallel set of sentences in the languages in which the system has been trained, we can generate $e_x$ and $e_y$. Both encodings, coming from a parallel test, have the same number of vectors each of which has the same dimensionality.

Our proposed analysis consists of inferring one of the decoders in the system ($d_x$ and $d_y$) using $e_x$ and $e_y$ as the input. This generates two different outputs: an autoencoding output and a machine translation output. As we have parallel references for both languages we can measure the BLEU score (Papineni et al., 2002) of each of the results against the reference to measure how the models perform.

Additionally, we can calculate a new BLEU score comparing the outputs of the autoencoding and the machine translation outputs. In the ideal case, encoders from two different languages have to produce the same representation for the same sentences. Therefore, the difference between the BLEU scores obtained from the autoencoding output and the translation output show how different the $e_x$ and $e_y$ representations are in terms of how the decoder is able to generate accurate results from them. Our analysis consists of evaluating the BLEU scores using the autoencoding output as a reference and the machine translation output as a hypothesis (this comparison is referred to as A-T). Figure 4 shows the full pipeline of this procedure.

Table 6 shows that the quality of the output of the decoder is much better when the input comes from an encoder of the same language (autoencoder) than from

<table>
<thead>
<tr>
<th>Decoder</th>
<th>Autoencoder</th>
<th>Machine translation</th>
<th>A-T</th>
</tr>
</thead>
<tbody>
<tr>
<td>EN</td>
<td>90.50</td>
<td>14.82</td>
<td>14.91</td>
</tr>
<tr>
<td>TR</td>
<td>96.20</td>
<td>12.56</td>
<td>14.56</td>
</tr>
</tbody>
</table>

*Note: The third column is the BLEU between autoencoder and translation outputs.*

**FIGURE 5** Encoder representations of the multi-way parallel 381 sentences. Turkish sentence representations (blue) compared to English sentence representations (red) and Kazakh sentences (green) [Color figure can be viewed at wileyonlinelibrary.com]
We also included the BLEU score between both the autoencoder and translation outputs (A-T), which is the measure that we are proposing to evaluate the quality of our intermediate representation. Low BLEUs in A-T indicates that we are still far from being able to decode from the common intermediate representation.

### 6.2 Visualization and translation examples

In the following, we analyze the intermediate representations at the last attention block of the encoders, where we force the similarity. To graphically show the presentation, we use an in-house visualization tool (Escolano, Costa-Jussà, Lacroux, and Vázquez, 2019) that is freely available. The tool trains a UMAP (McInnes, Healy, Saul, and Grossberger, 2018) model combining the representations of languages that reduces the dimensionality of the sentence representations.

In the following, we used the 381 multiway parallel sentences extracted specifically for this analysis (see the statistics in Table 1). Figure 5 shows the sentence representations created by their encoders. The separated clusters show that languages are not yet represented in the same space. A related work (Arivazhagan et al., ) shows similar results for the case of a multilingual system with shared encoder and decoder. While the system is able to produce compatible representations clear clusters can be observed for each language in the system. Plausible explanations for this difference may be the distance measure that we are using and/or the alignment of the source sentences. Some distance measures cause the representations to collapse in a small region of the space making them non-informative for the decoder. Our distance measure, the correlation distance, while it enforces the representations to correlate, it does not constrain the scale of the values in the contextual vectors. This measure enforces the sentence distribution within the same language to be similar between all languages. However, since we are not constraining the scale, each language can be represented in a different space region.

Table 7 shows some examples of English translations (using the English decoder) when using the three different encoders (English, Turkish, and Kazakh) together with the reference translation.

### 7 CONCLUSIONS

We proposed a novel translation architecture that aims at a common intermediate representation for the benefit of incremental training in machine translation.

While there are already some machine translation systems where the implicit emergence of an internal interlingua representation is suggested, the main proposed difference in the current paper is that we force the NMT system to learn an intermediate multilingual representation while using independent encoders and decoders. This is achieved by combining the maximum likelihood loss, normally used in NMT, together with an extra loss term that computes a measure of the distance between intermediate representations in different languages.

By achieving an interlingua representation, encoders and decoders are decoupled by having the interlingua as an interface. This leads to enabling every possible combination of encoder and decoder, effectively turning the quadratic needs, in terms of the training data and
resources, into linear ones. Furthermore, this decoupling also allows training encoders/decoders to/from a new language that has only parallel data with one of the already supported languages, enabling the translation to/from any of the other supported languages. As a consequence, our architecture allows us to extend the bilingual system to a multilingual system through incremental training. This particularity allows for quickly deploying new systems when requiring new languages into the system, which is highly relevant in scenarios such as humanitarian crisis.

Beyond the fact that our model presents a flexible architecture that enables scaling to new languages without retraining languages in the system, our model outperforms current bilingual and pivoting systems in the low-resource setting.

One of the next steps will be to exploit monolingual data in our architecture further avoiding a dependency on the availability of parallel data.

ACKNOWLEDGMENTS
This work also is supported in part by the Google Faculty Research Awards and, also by, the Spanish Ministerio de Economía y Competitividad, the European Regional Development Fund, the Agencia Estatal de Investigación through the postdoctoral senior grant Ramón y Cajal and the projects EUR2019-103819, PCIN-2017-079 and PID2019-10759RB-I00.

ENDNOTES
1 http://www.statmt.org/wmt19/
2 http://www.statmachine-translation.org/WMT17/
3 http://www.statmt.org/WMT19/
4 http://www.statmt.org/WMT19/
5 The datasets that we prepared for Kazakh-Turkish and Turkish-Kazakh-English, which are the only ones that not belong to a benchmark, are freely available under request.
6 Release v0.6.0 available at https://github.com/pytorch/fairseq
7 http://matrix.statmt.org/
8 https://github.com/elorala/interlingua-visualization

REFERENCES
Bahdanau, D., Cho, K., & Bengio, Y. (2015). Neural machine translation by jointly learning to align and translate. ICLR. San Diego, California, USA.


Lewis, W. D. (2010). Haitian creole: How to build and ship an mt engine from scratch in 4 days, 17 hours, & 30 minutes. EAMT. Saint-Raphaël, France.


Sutskever, I., Vinyals, O., & Le, Q. V. (2014). *Sequence to sequence learning with neural networks*. NIPS. Montréal, Canada.


**How to cite this article:** Escolano C, Costa-Jussà MR, Fonollosa JAR. From bilingual to multilingual neural-based machine translation by incremental training. *J Assoc Inf Sci Technol*. 2020; 1–14. [https://doi.org/10.1002/asi.24395](https://doi.org/10.1002/asi.24395)