Zero-Shot Translation for Indian Languages with Sparse Data

Giulia Mattoni giuliam@kantanmt.com
Pat Nagle patn@kantanmt.com
Carlos Collantes carlosc@kantanmt.com
Dimitar Shterionov dimitars@kantanmt.com
KantanMT.com, INVENT Building, Dublin City University Campus, Dublin 9, Dublin, IRELAND

Abstract

Neural Machine Translation (NMT) is a recently-emerged paradigm for Machine Translation (MT) that has shown promising results as well as a great potential to solve challenging MT tasks. One such a task is how to provide good MT for languages with sparse training data. In this paper we investigate a Zero Shot Translation (ZST) approach for such language combinations. ZST is a multilingual translation mechanism which uses a single NMT engine to translate between multiple languages, even such languages for which no direct parallel data was provided during training.

After assessing ZST feasibility, by training a proof-of-concept engine ZST on French↔English and Italian↔English data, we focus on languages with sparse training data. In particular, we address the Tamil↔Hindi language pair. Our analysis shows the potential and effectiveness of ZST in such scenarios.

To train and translate with ZST engines, we extend the training and translation pipelines of a commercial MT provider – KantanMT – with ZST capabilities, making this technology available to all users of the platform.

1 Introduction

Nowadays Machine Translation (MT) is an essential tool for the translation industry. The most used MT paradigms are Phrase-based Statistical Machine Translation (PBSMT) (Koehn et al., 2007) and Neural MT (Bahdanau et al., 2014; Sutskever et al., 2014; Cho et al., 2014). While PBSMT has been the state-of-the-art both in academia and industry for the last decade, recently NMT has showed great potential and in many cases has surpassed PBSMT (Bentivogli et al., 2016; Junczys-Dowmunt et al., 2016; Chung et al., 2016; Shterionov et al., 2017).

NMT, similar to PBSMT, is a data-driven MT paradigm, making it strongly dependent on the parallel data used for training. That is, the translation quality of an NMT system correlates with the quality and quantity of the training corpora. Freely accessible parallel corpora are available from various providers, such as: Opus[1], DGT-EC (European Commission)[2] and Linden/Clarin repository.[3] Within the industry, MT

[1]http://opus.lingfil.uu.se/
[2]https://ec.europa.eu/jrc/en/language-technologies/dgt-translation-memory
[3]https://lindat.mff.cuni.cz/repository/xmlui/handle/11858/00-097C-0000-0023-625F-0

systems are typically built with proprietary data – i.e., data with restricted access, provided by a translation vendor and tailored towards specific translation task(s) mainly because of data confidentiality requirements and/or because the data includes terminology and style that are specific for the translation task. Often, to build a custom MT system (i.e., an MT system that is customised according to a translation vendor's requirements) that can produce high-quality translations, proprietary and non-proprietary data are concatenated.

For some language pairs, however, there is not enough available parallel data (proprietary or non-proprietary) to build MT systems of high quality to meet users' requirements. This specifically applies to minority or low-resource languages – languages that have a low population density, are under-taught or have limited written resources or are endangered – as well as to language pairs **with sparse training data** – language pairs for which there have not been documented (human) translations that can be used as training data. Sparsity of training data for language pairs such as, e.g. Tamil or Hindi, which by itself are not low-resource languages, is a phenomenon that hinters the MT industry.

Aiming to overcome the data sparsity within the spectrum of the Indian languages, this paper investigates a zero-shot translation (ZST) (Johnson et al., 2016) strategy for the Hindi and Tamil languages. We built ZST engines on available parallel data to and from English (we use English as an intermediate or a pivot language).

To determine the viability and potential of ZST we first build a proof-of-concept (POC) ZST engine for high-resource languages (English, Italian, Spanish and French). Second, we build a ZST engine for Hindi and Tamil as well as Hindi and Tamil, using parallel corpora with English, Hindi and Tamil data as well as for English, Hindi and Tamil data to prove the applicability of ZST for sparse-data language pairs.

To determine the quality of these engines we compare their outputs with the results of (i) one-to-one NMT engines for the same language combinations and (ii) via a pivoting language. In particular, case (ii) boils down to using two different NMT engines – one that translates from the investigated source language into English and another that translates from English to the investigated target language.

We use the KantanMT[4] platform to train and translate ZST engines. KantanMT is a custom Machine Translation (MT) platform that allows its users to build custom MT systems covering more than 75 languages. The analysis of the resulting quality is performed through comparison of quality evaluation metrics and the A/B testing interface of the KantanMT platform (KantanLQR™). As a provider of commercial MT solutions, the KantanMT platform is designed and tailored to train and deploy one-to-one translation engines (Phrase-based Statistical Machine Translation and NMT). The research and development of ZST engines imposes certain architectural requirements to the KantanMT platform. In this work, we also discuss the changes that such a platform requires in order to accommodate ZST technology.

The main contribution of this paper is two-fold: on the one hand it is the insights that we draw from our analysis of ZST as a means to tackle the problem of sparse data; on the other hand, we extend the pipeline of a commercial MT – KantanMT.com – making ZST available to KantanMT users.

This paper is organised as follows. In Section 2 we present relevant background and motivate our work; in Section 3 we discuss our MT training and translation pipeline, the changes we have done in order to accommodate ZST technology and we outline the data used to build our ZST engines; Section 4 is devoted to the analysis of the

[4]`www.kantanmt.com`

translation capabilities of these engines; we conclude our work and present our future plans in Section 5.

2 Background and Motivation

2.1 Zero-shot Translation

Zero-shot translation (ZST) (Johnson et al., 2016) is an approach to train a single NMT engine to translate between multiple languages. Such a multilingual engine can translate from a source to a target language without having seen explicit parallel corpora for that specific language pair during training. ZST exploits transfer learning to overcome the need of building one-to-one translation engines. According to (Johnson et al., 2016), an NMT engine can be trained as a multilingual ZST engine[5] by simply augmenting the training data with a token before each segment stating the target language. In particular, a sentence S^{L1} in language $L1$ aligned to a sentence S^{L2} in language $L2$ will be augmented with a token <2$L2$>. Following their findings we exploit a similar approach to augment each segment of the parallel training corpora with a token to indicate the target language. Moreover, we extend this data processing step to handle different tokenisation rules for each language correctly. We add one more token to indicate the language of origin[6] of the specific sentence that will be used during tokenisation.

In the work of (Johnson et al., 2016; Ha et al., 2016) a single shared attention mechanism and a single 'universal' encoder-decoder across all languages is used. Firat et al. (2016) also present a multilingual approach that uses a shared attention mechanism. However, they use multiple encoders/decoders for each source and target language. Aiming at smallest possible alterations of our training and translation pipelines we focus on the single encoder/decoder model with shared attention. Such an architecture does not impose any changes to our platform (i.e. KantanMT), except in the preprocessing (both before training and before translation) step.

In (Johnson et al., 2016), the authors prove that mixing language pairs with little and large available data into a single multilingual NMT model produces a considerable translation quality improvement of the low resource language. This translation capabilities are due to the fact that all the parameters of the multilingual model are implicitly shared by all the language pairs. The analysis on multilingual NMT and zero-shot (or zero-resource) translation, given by (Firat et al., 2016), investigates multiple strategies for multi-way, multilingual translation engines. They show that an NMT engine trained on parallel data without data between two languages translates very poorly for these two languages. In contrast, adding pseudo-parallel data for these two languages to finetune the engine improves significantly the quality. They, also, investigate a more basic multilingual NMT engine – trained on two parallel corpora (with or without a finetuning corpus) and is focused to translate between two of these language pairs, in contrast to (Johnson et al., 2016) where the focus is on translating a plethora of languages with one engine.

Motivated by the promising results documented in the aforementioned publications, our main objective is to demonstrate that ZST is particularly beneficial when it comes to MT for language combinations with sparse parallel corpora. We aim to translate one particular language pair (Tamil$\rightarrow$Hindi) with a single encoder-decoder with shared attention mechanism NMT engine while using English$\leftrightarrow$Tamil and English$\leftrightarrow$Hindi as

[5]In the remaining of this paper we refer to multilingual NMT engines that are trained according to the Zero Shot Approach as *ZST engines.*

[6]We refer to the language of a specific sentence as its *language of origin* to differentiate between source and target languages.

well as a small set of Tamil↔Hindi data.

2.2 Indian languages

Research, conducted on MT for Indian languages, mainly focuses on to- and from-English translation (Sindhu and Sagar, 2016; Antony, 2013). In the survey of Antony (2013) of MT systems for Indian languages there is only one Tamil-Hindi system.[7] Even exploiting data-driven MT paradigms (such as PBSMT or NMT) that ease the creation and exploitation of MT systems even by non-linguistically informed users, the lack of parallel data is what restricts high-quality MT systems to be built. Ramasamy et al. (2012) present an English-Tamil PBSMT engine as well as a corpus of circa 200000 parallel sentences. Post et al. (2012) present parallel corpora for six Indian languages and English. Bojar et al. (2014) discuss the HindEnCorp dataset which constitutes of approximately 300000 parallel sentences. Another source for data are platforms like Opus and EMILLE. These resources, however are not sufficient (both quantity-wise as well as quality-wise) to build an efficient, domain oriented one-to-one MT engine between two Indian languages.

The aforementioned issues impose a translation gap between Indian languages. We exploiting ZST methodology in order to reduce this gap. We use various available parallel corpora, which we cleansed and organised, to training our ZST engines.

3 Zero Shot Translation Engines

3.1 Pipeline

The KantanMT platform has two main pipelines: one to train an MT engine and a second one to translate text with a selected MT engine. Figure 1 illustrates these pipelines.

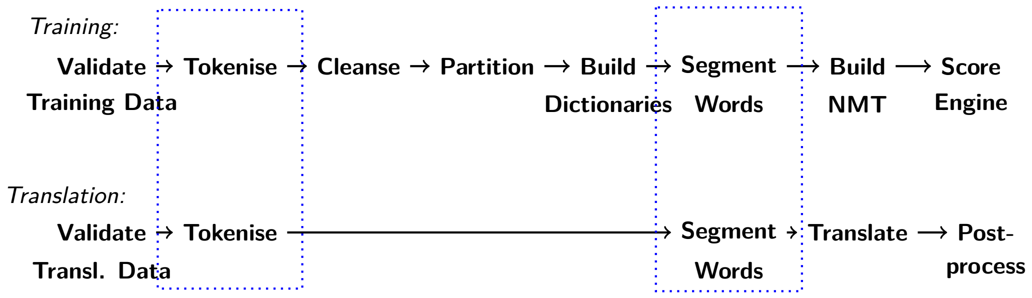

Figure 1: Abstract representation of the training and translation pipelines. Blue boxes indicate processing steps that are common for both pipelines. The input of the training pipeline is source and target data; the input of the translation pipeline is text to translate.

While their core processing mechanisms are different, as shown in Figure 1 they both use the same *tokenisation* step, as well as *word segmentation*. In practice, in the latter step a dictionary is used that is created in the *Build dictionaries* step in the training pipeline; this dictionary is then stored and reused during translation again in the *word segmentation* step.[8]

[7]We refer the interested reader to the (Antony, 2013) for more information on the system.
[8]We present more details about word segmentation and dictionaries in Section 4.

In order to support both training and translating with a ZST engine, it is necessary to adapt these common steps (i.e., the tokenisation and the word segmentation) such that they meet the following requirements:

1. Training and test data are augmented with ZST tokens as defined in Section 2.
2. Different languages require different tokenisation rules which needs to be accommodated in the tokenisation step. That is, the training and test data sets would contain sentences in different languages (see Section 2). The tokeniser would required to know their language of origin and tokenise them according to language-specific rules.
3. Any ZST token is not affected by any consecutive preprocessing step.
4. During both training and translation the output of the neural network does not contain any ZST token.

To meet the first requirement the user needs to introduce ZST tokens for the source and the target data. The target data needs to be augmented with one ZST token, which indicates the *language of origin* of the data. E.g., if the target data is in English, each sentence needs the prefix $*zst\_en*$ (if a locale is specified, e.g., British English, the prefix is $*zst\_en\_gb*$). The source data, however, needs two ZST tokens: one to indicate the *language of origin* and another to indicate the target language. These have the same form as mentioned above with the first ZST token referring to the *language of origin* and the second one indicates the target language. Example 3.1 illustrates the source and target data, augmented with ZST tokens.

Example 3.1
Source (English, original): It helps for detachment of umbilical cord.
Souce (English, with ZST tokens): $*zst\_en * *zst\_hi*$ It helps for detachment of umbilical cord.
Target (Hindi, original): आपको ईमेल एलर्ट के लिए सबस्क्रइब किया गया है।
Target (Hindi, with a ZST token): $*zst\_hi*$ आपको ईमेल एलर्ट के लिए सबस्क्रइब किया गया है।

In order to meet requirements 2, 3 and 4, we modified the *Tokenisation* step as well as the *Word segmentation* step in our pipelines. The *Tokenisation* step is adapted to read the first from the two ZST tokens from each sentence of the *source* data and the only one ZST token from each sentence of the *target* data and extract the language and locale codes. Then it removes these ZST tokens. Next, each sentence will be tokenised according to tokenisation rules specific for the language and locale codes extracted from the ZST token.

The *Word segmentation* step, which is prior to the *Build NMT* step (in the training pipeline) or to the *Translation* step (in the translation pipeline) will split each word into subword units (Sennrich et al., 2016). During this step the ZST tokens may become segmented which would negatively impact the training of the network. We augment the *Word segmentation* with an extra step to recover any segmented ZST token.

Example 3.2 shows the form of the source and the target data prior to training. The @@ symbols are used as a delimiter for the word segmentation.

Example 3.2
Source (English, original): You have been subscribed to email alerts .
Source (English, tokenized, word-segmented): $*zst\_hi*$ You have been sub@@ scribed to email al@@ er@@ ts .

Target (Hindi, original): आपको ईमेल एलर्ट के लिए सबस्क्राइब किया गया है।
Target (Hindi, tokenized, word-segmented): आपको ईमेल एल@@ र्ट के लिए सब@@ स्क्र@@ ाइ@@ ब किया गया है ।

Figure 2 shows the changes that were introduced to our pipelines.

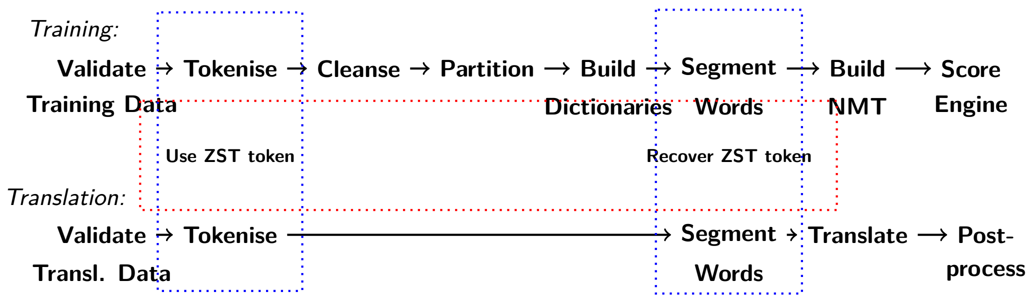

Figure 2: Abstract representation of the training and translation pipelines augmented with additional functionalities required to accommodate ZST. Blue boxes indicate processing steps that are common for both pipelines. The red boxes indicate the additional steps that are required for ZST. The input of the training pipeline is source and target data with ZST tokens; the input of the translation pipeline is text to translate with ZST tokens.

3.2 Engines

With the adapted pipelines we can now easily build ZST engines and use them to translate between language pairs for which parallel data was not provided. In particular, given parallel data set between languages $L1$ and $L2$ as well as between $L2$ and $L3$ we can build a ZST engine that translates a text in $L1$ into $L3$.

Example 3.3 *Consider we have available parallel data between English (EN) and Tamil (TA) and between English and Hindi (HI). We use TA and HI data both as source and as target (aligned correctly with their EN counterpart), and the same for the EN data (aligned correctly with the TA and HI) and train a ZST engine:*

Source	*Target*
English	*Tamil*
Tamil	*English*
English	*Hindi*
Hindi	*English*

This engine would allow us to translate from TA to HI, but also the other way round – from HI to TA. Moreover, it would translate from EN to HI or TA (and vice-versa) as well as from EN to EN.

Example 3.3 shows how we use the available parallel data both as source and as target, aligned correctly, in order to train a basic ZST engine. In general, given data for N languages all aligned with 1 other language (in Example 3.3 that is English) we can build a ZST engine to translate between all of the $N * (2 + N)$ source and target options, including (as in Example 3.3) translating between the same language.

The reason that a ZST engine requires the data to be used both as source and as target is that the neural network will learn to map unseen language pairs through their

Engine Name:	Languages:	Number of Sentences:	Source Words:	Target Words:	Used to translate:	Domain
ZST_1	EN↔FR, EN↔IT	798 996	15 075 691	15 075 789	FR→IT	Legal
$Pivot_1$	EN→FR	198 999	3 844 982	3 475 693	EN→FR	Legal
$Pivot_2$	IT→EN	198 999	3 399 530	3 502 284	IT→EN	Legal
ZST_2	EN↔TA, EN↔HI	1 009 892	15 284 069	15 284 069	TA→HI	General
ZST_3	EN↔TA, EN↔HI, TA→HI	1 051 631	15 691 380	15 691 380	TA→HI	General, Technical
$Pivot_3$	TA→EN	168 871	2 759 734	3 960 123	TA→EN	General
$Pivot_4$	EN→HI	268 317	3 338 686	3 620 445	EN→TA	General
$one\text{-}to\text{-}one_1$	TA→HI	41 739	365 571	546 584	TA→HI	Technical

Table 1: Summary of the data used to build ZST and one-to-one engines.

common language.[9]

In the scope of this work we build ZST engines with English, French, Italian data, as well as with English, Tamil and Hindi data. First, we build a proof-of-concept ZST engine on English-French, English-Italian data; we use this engine to translate between French and Italian. To test the performance of this engine we also build two One-to-one engines: one from French to English and another from English to Italian. We refer the latter engines as *Pivot* engines and use them in a sequence to derive an Italian translation, starting from a French text.

Next, we focus on the Indian languages and build two ZST engines - one on English-Tamil and English-Hindi data and a second one on the same English-Tamil and English-Hindi data as well as Tamil-Hindi data. Then we build three one-to-one engines: one Tamil-Hindi, one Tamil-English and a third one English-Hindi all using the same data as for the ZST engines.

Table 3.2 enumerates the available data and the engines we trained.

In Section 4 we present and discuss our findings from comparing the translation quality of these engines.

4 Experiments

We perform our analysis on the MT engines – ZST or one-to-one – enumerated in Table 3.2.

NMT setup. Our training and translation pipelines are based on the OpenNMT toolkit[10] (Klein et al., 2017) version 0.7. As learning optimizer we use ADAM (Kingma and Ba, 2014) with learning rate 0.0005. We train our networks for at least 5[11] epochs on NVIDIA G520 GPUs with 4GB RAM (each model is trained on a single GPU). The maximum batch size if 50. The maximum input length used for training is 150.

Dictionaries. Each NMT engine is trained on two dictionaries – one for the source and one for the target data. For ZST engines, we use the concatenated source or target training data to build a source or target dictionary. The dictionaries are composed of word segments in order to increase the vocabulary capabilities of the network and avoid out-of-vocabulary (OOV) problems. We use byte pair encoding (BPE) Sennrich et al. (2016) of 40 000 operations to build the word segments.[12] We prepare the dictionaries from normal-cased (i.e., lower- and upper-cased) tokenised data.

[9] For more details we refer the interested reader to (Johnson et al., 2016).

[10] `http://opennmt.net/`

[11] We present and analyse results of engines with the same number of epochs as to make the comparison fair.

[12] For data in Chinese, Japanese, Korean or Thai, our pipelines use dictionaries based on character-by-character segmentation (Chung et al., 2016). That is, each word segment in the dictionary is a single character. BPE is used for all other languages, including Tamil and Hindi.

Engine:	BLEU*	F-Measure*	Perplexity**
ZST_2	0.21	3.26	17.12
ZST_3	9.78	26.40	21.91
one-to-one_1	8.20	22.16	78.96
$Pivot_3$ + $Pivot_4$	0.16	16.94	24.85

Table 2: Evaluation of our Indian engines. * - the higher the better; ** - the lower the better.

Result analysis. We began our experiments using a ZST engine consisting of Legal domain data acquired from the European Commission – DGT, which is freely available for use. We decided on a POC engine consisting of English↔French and English↔Italian parallel data sets. We also constructed two one-to-one engines for the same language pairs as the ZST (i.e., $Pivot_1$ and $Pivot_2$, see Table 3.2). We started by running 50 sentences of legal domain content that the engines had not seen during training. The translation test set content was in French and needed to be translated into Italian. First, the ZST engine translated the content from French to Italian. Next the same French legal content was translated through the French↔English engine ($Pivot_1$); then we used the output from this engine as input for the English↔Italian engine ($Pivot_2$).

We then evaluated both Italian outputs produced by the ZST_1 and $Pivot_2$ engines running an A/B testing with KantanLQR, KantanMT's quality evaluation platform. A native Italian speaker with French fluency ranked the translations. The result of the A/B test was conclusively in favour of ZST, with our reviewer choosing 58 percent of the test segments from this engine as better quality than that of the pivot engines. With this result from our POC engine with high resource languages we began experimenting with low resource languages, in particular English↔Tamil and English↔Hindi.

The initial translation tests for our ZST Indian language engine were not as promising as we had hoped from the results of our POC engines. The output was not a complete translation to Hindi but a combination of all 3 input languages of English, Tamil and Hindi. From this result we concluded that we would need more parallel data in both language pairs and possibly aligned data for Tamil↔Hindi to help bridge the sparse data gap. We augmented our test data (the statistics of our data to built the ZST_3 engine, shown in Table 3.2).

We use BLEU (Papineni et al., 2002) and F-Measure (Melamed et al., 2003) to assess the quality of the Indian engines. We also report the perplexity of the engine scored after training is finished. To test whether indeed ZST can improve on one-to-one or pivot engines, we use the same test data set. It contains 500 sentences that are from the same domain of the one-to-one engine (one-to-one_1 in Table 3.2). Our results are summarised in Table 4.

While the enlisted scores for the given test set are in general very low, we observe that the best scores are achieved by the ZST_3 engine – the ZST engine which combines parallel data in different languages and a small set of Tamil↔Hindi data – the BLEU and F-Measure scores for the ZST_3 engine are the highest.

Furthermore, these results confirm that a ZST engine with parallel data for the languages of interest can significantly boost the translation capabilities (compare the scores of ZST_2 and ZST_3).

We ought to note that while these engines may not produce high-quality Tamil→Hindi translations (according to the evaluation metrics reported in Table 4) they show that ZST has a potential and deserves further investigation. Our direct efforts are in bringing a Tamil→Hindi engine together with other Indian languages to industry standards.

5 Conclusions and Future Work

In this paper, we present our first Zero Shot Translation engines for languages with sparse training data. We observed that while ZST produces good quality output for high resource languages (with good training data), it is not performing as good for the Tamil↔Hindi language pair that we used as out main use case. However, our ZST engine that combines multiple-source data and Tamil↔Hindi performs better than the rest of the Indian engines.

Our results showed that further experiments on zero shot translation are needed. First, we will focus on data analysis in order to understand which data combinations are useful for ZST and which are not. Next, we intend to test ZST for other language combinations in order to evaluate which language families or specific languages could benefit the most from such a translation approach.

In addition, with this work we adapted the training and translation pipelines of a commercial MT provider to support ZST engines. In the future we aim to further improve these pipelines and provide more and better ZST services to the users.

References

Antony, P. J. (2013). Machine translation approaches and survey for indian languages. *IJ-CLCLP*, 18(1).

Bahdanau, D., Cho, K., and Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. *CoRR, Accepted for oral presentation at the International Conference on Learning Representations (ICLR) 2015*, abs/1409.0473.

Bentivogli, L., Bisazza, A., Cettolo, M., and Federico, M. (2016). Neural versus phrase-based machine translation quality: a case study. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*.

Bojar, O., Diatka, V., Rychlý, P., Stranak, P., Suchomel, V., Tamchyna, A., and Zeman, D. (2014). Hindencorp – hindi-english and hindi-only corpus for machine translation. In Chair), N. C. C., Choukri, K., Declerck, T., Loftsson, H., Maegaard, B., Mariani, J., Moreno, A., Odijk, J., and Piperidis, S., editors, *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, Reykjavik, Iceland. European Language Resources Association (ELRA).

Cho, K., van Merriënboer, B., Gülçehre, Ç., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. (2014). Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of EMNLP 2014*, Doha, Qatar. Association for Computational Linguistics.

Chung, J., Cho, K., and Bengio, Y. (2016). A character-level decoder without explicit segmentation for neural machine translation. In *Proceedings of the ACL*, Berlin, Germany.

Firat, O., Sankaran, B., Al-Onaizan, Y., Yarman-Vural, F. T., and Cho, K. (2016). Zero-resource translation with multi-lingual neural machine translation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, pages 268–277.

Ha, T., Niehues, J., and Waibel, A. H. (2016). Toward multilingual neural machine translation with universal encoder and decoder. In *Proceedings of the Thirteenth International Workshop on Spoken Language Translation (IWSLT '16)*, Seattle, WA, USA.

Johnson, M., Schuster, M., Le, Q. V., Krikun, M., Wu, Yonghui Chen, Z., and Thorat, N. (2016). Google's multilingual neural machine translation system: Enabling zero-shot translation.

Junczys-Dowmunt, M., Dwojak, T., and Hoang, H. (2016). Is neural machine translation ready for deployment? A case study on 30 translation directions. *CoRR*, abs/1610.01108.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Klein, G., Kim, Y., Deng, Y., Senellart, J., and Rush, A. M. (2017). OpenNMT: Open-Source Toolkit for Neural Machine Translation. *ArXiv e-prints*.

Koehn, P., Hoang, H., Birch, A., Callison-Burch, C., Federico, M., Bertoldi, N., Cowan, B., Shen, W., Moran, C., Zens, R., Dyer, C., Bojar, O., Constantin, A., and Herbst, E. (2007). Moses: Open Source Toolkit for Statistical Machine Translation. In *Proceedings of ACL 2007, demonstration session*, Prague, Czech Republic.

Melamed, I. D., Green, R., and Turian, J. P. (2003). Precision and Recall of Machine Translation. In *NAACL-HLT*.

Papineni, K., Roukos, S., Ward, T., and Zhu, W.-J. (2002). BLEU: A Method for Automatic Evaluation of Machine Translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, Stroudsburg, PA, USA.

Post, M., Callison-Burch, C., and Osborne, M. (2012). Constructing parallel corpora for six indian languages via crowdsourcing. In *Proceedings of the Seventh Workshop on Statistical Machine Translation*, pages 401–409, Montréal, Canada. Association for Computational Linguistics.

Ramasamy, L., Bojar, O., and Žabokrtský, Z. (2012). Morphological processing for english-tamil statistical machine translation. In *Proceedings of the Workshop on Machine Translation and Parsing in Indian Languages (MTPIL-2012)*, pages 113–122.

Sennrich, R., Haddow, B., and Birch, A. (2016). Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*.

Shterionov, D., Nagle, P., Casanellas, L., Superbo, R., and O'Dowd, T. (2017). Empirical Evaluation of NMT and PBSMT Quality for Large-scale Translation Production. In *EAMT*.

Sindhu, D. and Sagar, B. (2016). Study on machine translation approaches for indian languages and their challenges. In *Electrical, Electronics, Communication, Computer and Optimization Techniques (ICEECCOT), 016 International Conference on*, pages 262–267. IEEE.

Sutskever, I., Vinyals, O., and Le, Q. V. (2014). Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*.