Interspeech 2018
2-6 September 2018, Hyderabad

Fast Derivation of Cross-lingual Document Vectors from Self-attentive Neural Machine Translation Model

Wei Li and *Brian Mak*

Department of Computer Science and Engineering
The Hong Kong University of Science and Technology
{wliax, mak}@cse.ust.hk

Abstract

A universal cross-lingual representation of documents, which can capture the underlying semantics is very useful in many natural language processing tasks. In this paper, we develop a new document vectorization method which effectively selects the most salient sequential patterns from the inputs to create document vectors via a self-attention mechanism using a neural machine translation (NMT) model. The model used by our method can be trained with parallel corpora that are unrelated to the task at hand. During testing, our method will take a monolingual document and convert it into a "**N**eural machine **T**ranslation framework based cross-lingual **D**ocument **V**ector" (NTDV). NTDV has two comparative advantages. Firstly, the NTDV can be produced by the forward-pass of the encoder in the NMT, and the process is very fast and does not require any training/optimization. Secondly, our model can be conveniently adapted from a pair of existing attention-based NMT models, and the training requirement on parallel corpus can be reduced significantly. In a cross-lingual document classification task, our NTDV embeddings surpass the previous state-of-the-art performance in the English-to-German classification test, and, to our best knowledge, it also achieves the best performance among the fast decoding methods in the German-to-English classification test.

Index Terms: cross-lingual text classification, distributed representation, neural machine translation model

1. Introduction

Distributed representation of text has been one of the most popular topics in natural language processing. Comparing to text features in discrete spaces, distributed representations can include richer syntactic and semantic information [1, 2, 3, 4, 5]. By embedding words, sentences or documents into a continuous space, the syntactic relationship, semantic similarity, topic/gene categories or other high-level linguistic information between the embedded texts can be represented through their relative coordinates in the vector space. Therefore, the distributed representation can help us discover the relationship between words and sentences, categorizing documents and sharing the knowledge between related texts [1, 2].

Recently along this endeavour, research on cross-lingual distributed representation of multilingual texts has received increasing attention. A cross-lingual embedding of the texts from different languages to a unified space will enable comparison and sharing knowledge between languages [6, 7, 8, 9, 10, 11]. In many applications, we often have very little data from users speaking/writing in a different language. On the other hand, we often have enough parallel sentences from unrelated corpora. Thus, with cross-lingual embedding, we can use the unrelated parallel texts to help us process and understand the resource-scarce languages in such scenarios.

The existing research can be categorized into cross-lingual word embeddings [12, 13, 7, 9] and cross-lingual text sequence (document/sentence) embeddings [6, 14, 8]. For cross-lingual representation of text sequence, there are several popular approaches. One is the multi-lingual extension of the original paragraph vector [14, 9]. In this approach, a model is firstly trained on a parallel corpus, then the document vector (as the only free parameters) is trained on the monolingual test data. One example is the $para\_doc$ which is currently the state-of-the-art cross-lingual document/sentence vector, achieving the best result in the cross-lingual document classification task on the Reuter's RCV1/RCV2 dataset[12, 15]. The other popular approaches include the extension of the multilingual/multi-task sequence to sequence/vector learning frameworks [6]. The document/sentence vector can be some intermediate state or the output of the model (often with a penalty term to minimize the distance between the vectors of the input sentence pair). After having been trained on some parallel corpus, the document or sentence vector of the test data can be obtained by a forward-pass running on the trained model.

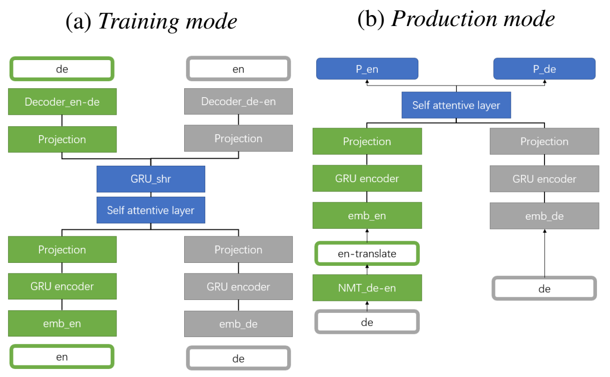
In this study, we present a neural machine translation (NMT) based cross-lingual self-attentive document/sentence vectorization model (NTDV) to convert a text sequence of variable length into a fix-sized vector. The model is similar to the line of works that uses the multilingual/multi-task sequence to sequence learning framework. The NTDV model has the following characteristic: Firstly, as it only uses the forward-pass in the production mode, it is faster than methods that require training/optimization in producing document vectors. For the same reason, this model does not need validation/tuning of the hyper-parameters in the production mode, and will always produce consistent vectors given the trained model. (The training/optimization of the first approach is often sensitive to the learning rate and parameter initialization scheme in the production mode). Secondly, the NTDV model reuses the knowledge from NMT models and the training time is reduced significantly. Finally, in the Reuter's RCV1/RCV2 cross-lingual document classification task, the NTDV model achieves the best result among the fast vectorization methods (which do not require training in production mode), and it also surpasses the current state-of-the-art method in the English-to-German classification test.

2. Model architecture

2.1. The attention-based NMT model

Our NTDV model is built upon the attention-based NMT framework [16], which adopts an encoder-decoder structure with an

10.21437/Interspeech.2018-1459


(a) *Training mode*

(b) *Production mode*

Figure 1: *The cross-lingual self-attentive document/sentence vectorization model.*

attention mechanism that is coupled with a conditional GRU ($cGRU_{att}$) layer [16]. The attention mechanism allows the decoder to *pick* the most salient information from the encoder according to the decoder state at each step. The $cGRU_{att}$ is a deep GRU with two coupled recurrent units at each time steps. Together with the attention mechanism, it allows better handling of information from the past (previous outputs) and from below (attention contexts). The encoder is composed of a word embedding layer for the input language tokens and a bi-directional GRU layer, whereas the decoder is composed of a word embedding layer for previous output tokens and a $cGRU_{att}$ layer.

The encoder and decoder in the model is connected by a fully connected layer, whose input is the encoder's hidden states over all input tokens and whose output sets the initial state of the conditional GRU in the decoder. The encoder also feeds information to the decoder through an attention mechanism. The decoder state $\boldsymbol{s}_j$ for generating the jth output token is determined by its previous state $\boldsymbol{s}_{j-1}$, its previous output token y_{j-1} and the context vector $\boldsymbol{c}_j$ from the attention layer as follows:

$$\boldsymbol{s}_j = cGRU_{att}(\boldsymbol{s}_{j-1}, y_{j-1}, \boldsymbol{c}_j) \ . \tag{1}$$

Let L_x be the length of an input sequence. The encoder output, which is the collection of the hidden states of its last layer from all L_x frames can be represented by the matrix $\mathbf{H} = [\boldsymbol{h}_1 \cdots \boldsymbol{h}_i \cdots \boldsymbol{h}_{L_x}]$. Then $\boldsymbol{c}_j$ is computed by the attention mechanism (ATT) as a weighted mean of the encoder's hidden states $\mathbf{H}$:

$$\boldsymbol{c}_j = ATT(\mathbf{H}, \hat{\boldsymbol{s}}_j) = \sum_{i}^{L_x} \alpha_{ij} \boldsymbol{h}_i \tag{2}$$

$$\alpha_{ij} = \frac{exp(e_{ij})}{\sum_{n=1}^{L_x} exp(e_{nj})} \tag{3}$$

$$e_{ij} = \boldsymbol{v}_a^T tanh(\mathbf{U}_a \hat{\boldsymbol{s}}_j{}^T + \mathbf{W}_a \boldsymbol{h}_i^T) \tag{4}$$

where $\hat{\boldsymbol{s}}_j$ is the intermediate hidden state in the conditional GRU; α_{ij} is the normalized alignment weight between the ith source token and the jth target token; $\mathbf{U}_a$, $\mathbf{W}_a$, $\boldsymbol{v}_a$ are the model parameters used in the attention mechanism.

2.2. The self-attentive cross-lingual document/sentence vectorization model

As pointed out by [17, 18], the information coming from below (the attention over the encoder's states) mainly helps to determine the word/sentence semantic information and content — the *adequacy*. On the other hand, the information coming from the past (historical information in decoder LSTM/GRU) generally contributes to the correct production of function words and word order — the *fluency*. This model focuses on extracting the adequacy information via a self-attentive layer, which summarizes the encoder information from an input text sequence of variable length into a fix-sized matrix as a primary form of the document/sentence representation by taking advantage of the selective characteristic of the self-attentive layer [19].

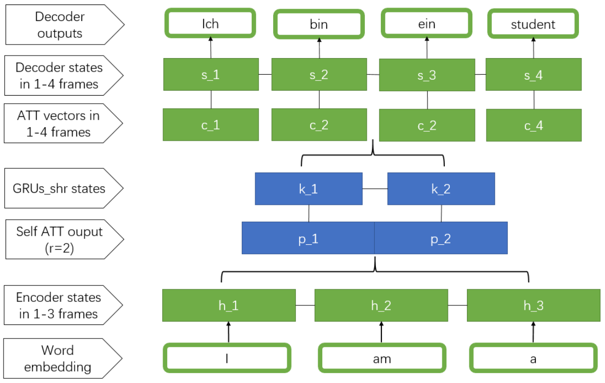


Figure 2: *The cross-lingual self-attentive document/sentence vectorization model (illustrated along time axis).*

Similar to the case of multilingual NMT model, we build the NTDV model by creating shared layers between NMT$_{a\rightarrow b}$ and NMT$_{b\rightarrow a}$, where $a \rightarrow b$ means translation from the source language a to the target language b. In details, we insert between the encoders and decoders of the NMT$_{a\rightarrow b}$ and NMT$_{b\rightarrow a}$ two shared layers: a self-attentive layer followed by a GRU layer (GRU_{shr}). By doing that, the shared layers forces the information encoded by the encoders of both languages to be represented in the same vector space, and thus resulting in a distributed cross-lingual document representation.

Figure 1(a) shows the NTDV model in the training mode, with a being English (en) and b being German (de). During training, given a pair of English and German sentences, the English sentence will be processed through the encoder of NMT$_{en\rightarrow de}$, the shared layers, and its decoder resulting in the cross entropy loss of cost$_{en\rightarrow de}$. Similarly, the German sentence in the pair will be processed through the encoder of NMT$_{de\rightarrow en}$, the same shared layers, and its decoder resulting in the cross entropy loss of cost$_{de\rightarrow en}$. The final loss of the NTDV model due to this sentence pair is the sum of $cost_{en\rightarrow de}$ and $cost_{de\rightarrow en}$.

Inspired by the structured self-attentive sentence embedding and the transformer network [19, 20], we use a self-attentive layer with multiple heads. Let $\boldsymbol{h}_i$ be the hidden state of the encoder, after the linear projection layer, it becomes

$$\hat{\boldsymbol{h}}_i = \boldsymbol{h}_i \mathbf{W}_e + \boldsymbol{b}_e \ , \tag{5}$$

where $\mathbf{W}_e$ is the weight matrix of the projection layer and $\boldsymbol{b}_e$ is the bias. Note that $\mathbf{W}_e$ is different for different language pairs. The self-attention vector $\boldsymbol{p}_m$ produced by the mth attention head is:

$$g_i^m = softmax\left(\frac{(\hat{\boldsymbol{h}}_i \mathbf{W}_q^m + \boldsymbol{b}_q^m)(\hat{\boldsymbol{h}}_i \mathbf{W}_k^m + \boldsymbol{b}_k^m)^T}{\sqrt{d_h}}\right) \tag{6}$$

$$\boldsymbol{p}_m = \sum_{i=1}^{L_x} g_i^m (\hat{\boldsymbol{h}}_i \mathbf{W}_v^m + \boldsymbol{b}_v^m) \tag{7}$$

where i is the index of input frames; m is the index of different attention head; $\hat{\boldsymbol{h}}_{\boldsymbol{i}}$ is the encoder state after the projection layer; $\mathbf{W}_q^m$, $\mathbf{W}_k^m$, $\mathbf{W}_v^m$ and $\boldsymbol{b}_q^m$, $\boldsymbol{b}_k^m$, $\boldsymbol{b}_v^m$ are model parameters of the mth attention head, where the subscripts q, k, v refer to quantities related to the *query*, *key* and *values* in the attention mechanism (as described in the transformer network[19]). In this study, although the same vectors are used for querys, keys and values, their projection weight matrices $\mathbf{W}_q^m$, $\mathbf{W}_k^m$, and $\mathbf{W}_v^m$ are different, and they all have the dimension $d_h \times d_h$, where d_h is the number of hidden units in the self-attentive layer and the shared GRU layer. $\boldsymbol{b}_q^m$, $\boldsymbol{b}_k^m$, and $\boldsymbol{b}_v^m$ are vectors of dimension d_h. Note also that compared with the self-attentive mechanism in the transformer network, $\hat{\boldsymbol{h}}_{\boldsymbol{i}}$ in our case does not attend to other encoder states at different time frames. The reason is that our goal is to summarize the input sequence to a fixed-length vector, and the self-attentive layer therefore focuses only on the overall sequential pattern as in the case of the structured self-attentive sentence embedding[20].

For an attention system with r heads, its r self-attention output vectors are grouped together in the context matrix $\mathbf{P} = \{\boldsymbol{p}_1, \ldots, \boldsymbol{p}_m, \ldots, \boldsymbol{p}_r\}$. If we set r to 1, then $\mathbf{P}$ will be a single context vector consisting of the weighted average information from the encoder. In general, the fixed-sized $\mathbf{P}$ is put through GRU_{shr} layer one row at a time to get an output from the GRU which is given by

$$\boldsymbol{k}_i = GRU_{shr}(\boldsymbol{p_i}, \boldsymbol{p_{i-1}}) \ , \tag{8}$$

where $\boldsymbol{p_i}$ is the ith row $\mathbf{P}$. $\boldsymbol{k}_i$ also goes through a projection layer as shown in Figure 1 before being input to the decoder part. The projected GRU_{shr} output is given by

$$\hat{\boldsymbol{k}}_{\boldsymbol{i}} = \boldsymbol{k_i}\mathbf{W}_p + \boldsymbol{b}_p \ , \tag{9}$$

where $\mathbf{W}_p$ is the projection weight matrix, and $\boldsymbol{b}_p$ is the bias. The projected GRU_{shr} outputs from all the r context vectors are grouped together in the matrix $\hat{\mathbf{K}} = \{\hat{\boldsymbol{k}}_{\boldsymbol{1}}, \ldots, \hat{\boldsymbol{k}}_{\boldsymbol{i}}, \ldots, \hat{\boldsymbol{k}}_{\boldsymbol{r}}\}$. $\hat{\mathbf{K}}$ is then fed to an attention mechanism similar to the attention layer in the aforementioned NMT model to give the context vector for the decoder:

$$\boldsymbol{c}_j = ATT(\hat{\mathbf{K}}, \hat{\boldsymbol{s}}_{\boldsymbol{j}}) \ , \tag{10}$$

where $\boldsymbol{c}_j$ and $\hat{\boldsymbol{s}}_{\boldsymbol{j}}$ are the same context vector and intermediate decoder state vector as in Eq. 2. In Figure 2, the input sentence has 3 words (or frames) and the output sentence has 4 words (or frames). The self-attentive layer has two heads (i.e., $r = 2$). No matter how long the sentence is, the self-attentive layer always selects and summarizes the information into the context matrix $\mathbf{P}$ of dimension $2 \times d_h$.

2.3. Deriving the document vectors in production mode

Figure 1(b) shows how the NTDV model works in the production mode when the input is a German document/sentence. In the production mode, only the encoder and the self-attentive layer is used, and either or both of the encoder of NMT$_{en\rightarrow de}$ and NMT$_{de\rightarrow en}$ may be used. If we want to use the encoder of NMT$_{en\rightarrow de}$, the German sentence is first translated to English using the NMT$_{de\rightarrow en}$, and the translated English output is then passed to the encoder of NMT$_{en\rightarrow de}$ to get the context vectors $\mathbf{P}^{(en)}$ (and $a \equiv en$). On the other hand, the encoder of NMT$_{de\rightarrow en}$ may be used directly to produce the desired context vectors $\mathbf{P}^{(de)}$ (and $b \equiv de$). In this paper, we propose the following different document/sentence representation vectors in the form of summation or concatenation of the rows in $\mathbf{P}^{(a)}$ and/or $\mathbf{P}^{(b)}$, where $\boldsymbol{p}_i^{(a)}$ and $\boldsymbol{p}_i^{(b)}$ are the ith row of $\mathbf{P}^{(a)}$ and $\mathbf{P}^{(b)}$, respectively:

$$\begin{aligned}
\text{NTDV}_a &= \sum_{i=1}^{r} \boldsymbol{p}_i^{(a)} && (11)\\
\text{NTDV}_b &= \sum_{i=1}^{r} \boldsymbol{p}_i^{(b)} && (12)\\
\text{NTDV}_{a:b} &= [\text{NTDV}_a, \text{NTDV}_b] && (13)\\
\text{NTDV}_{a+b} &= \text{NTDV}_a + \text{NTDV}_b \ . && (14)
\end{aligned}$$

The proposed document vectors may be produced by only a forward-pass through the model till the self-attention layer; no backward-propagation nor training is required. As a result, it is very fast and the costly computation of the softmax layer with a large vocabulary is spared. Before NTDV model training, the neural network translation models, NMT$_{a\rightarrow b}$ and NMT$_{b\rightarrow a}$, are first trained. Their parameters are used in the NTDV model without any re-estimation. NTDV model training only updates the parameters of the shared layers. In this way, we significantly reduce the training cost the NTDV model by transferring knowledge from the NMT model to the latter.

Table 1: *The dimension of various vectors.*

Feature	Dimension
NTDV$_{en}$	1024
NTDV$_{de}$	1024
NTDV$_{en+de}$	1024
NTDV$_{en:de}$	2048

3. Experiments

3.1. Training and text processing

We train our models on the Europarl v7 parallel corpus. The English (en) and German (de) pair of the corpus consists of about 1.9M parallel sentences with 49.7M English tokens and 52.0M German tokens [21]. We segment words via byte-pair encoding (BPE) [22, 23]. Unless otherwise stated, we follow the default training and text processing settings in the NEMATUS toolkit [16], upon which our codes are built.

In training the attention-based NMT models, we use the default setting with mini-batches of size 80, a vocabulary size of 85,000, a maximum sentence length of 50, word embeddings of size 500, and hidden layers of size 1024. The models are trained with the Adam optimizer [24] using the cross-entropy loss. Training stops when the BLEU score does not improve on the development set provided by the Workshop in Machine Translation [14]. In training the NTDV model, we use mini-batches of size 40 and set the number of heads r to 4. The model parameters of NTDV are initialized by the model parameters of the trained NMT model with the exception of the two newly added shared layers. The training procedure of the NTDV model is basically the same as that of the NMT model except that it always runs for five epochs with the original NMT parameters fixed, and early stopping with an evaluation set is not employed. In this way, we transfer the knowledge from the attention-based NMT models to our NTDV model and the training time of the NTDV model is greatly reduced. In translation, the attention-based NMT model in the pipeline also uses the default setting of the NEMATUS toolkit except that the beam size is set to 1 (to increase the translation speed).

3.2. Cross-lingual document classification (CLDC) on RCV1/RCV2

The effectiveness of the NTDVs is evaluated on the cross-lingual document classification task on Reuter's RCV1/RCV2 dataset [12, 15]. In this task, 1K English documents of four categories (Corporate/Industrial, Economics, Government/Social, and Market) are given to classify the category of the 5K German documents in the test set, and vice versa. The NTDV model will produce a document representation vector for each document in either language. As a document in the corpus usually consists of several sentences, there are two ways to produce the document vector. One method is to treat the entire document as a continuous sequence of words and uses the NTDV model to produce a single NTDV for the document. We will re-name such document vectors simply as DV for the sake of brevity. The other method is to produce one NTDV for each sentence, and then sum and average the sentence NTDVs together to form the final (sentence-based) document vector, which will be denoted as SV.

During production mode, the max length of the inputing sequence is set to be 500 for SV and 2000 for DV. Sequences longer than these thresholds would be cut off to save the memory space. A linear SVM classifier from scikit-learn [25] is trained on the document vectors of one language produced from the training set. The default settings of SVM training in the scikit-learn toolkit are used except that the maximum number of iterations is set to 5000 and the class weight is set to 'balanced'. After training, the SVM classifier will be used to classify the category of input document vectors of the opposite language in the test set.

4. Results

Table 2 shows the performance of various methods on the CLDC task. The method $para\_doc$ marked with * is not considered as a fast method in production mode as it would require parameter training during testing. Note that all the baselines are cited from the original publications; for those publications that report multiple results with and without using additional mono-lingual data, we cite the result that does not use additional data for a fair comparison. Among all the models, the BAE model [7] requires additional monolingual data (from the CLDC datasets) during training. MT_base is the machine translation baseline in the original study [12], while $para\_doc$ achieves the best performance currently. All the other existing methods fall very far behind $para\_doc$ in the de→en category. However, DVs and SVs produced by our proposed NTDV model give better performance than most of the existing methods with the only exception of $para\_doc$ which still performs better than ours in the de→en task. More specifically, our DV$_{en:de}$ has the best en→de classification performance and outperforms $para\_doc$ with a significant margin. In the de→en task, our SV$_{en:de}$ achieves the second best result that outperforms the third best also by a significant margin.

Our NTDV model has the advantage that it does not require validation and hyper-parameter tuning on the monolingual data at test time. It can be directly applied to the text in the CLDC task after the model has been trained on the parallel data. Note that our goal is to derive document/sentence vectors quickly, as we believe that it would be impractical to do adaptive training in many online systems. Although the NTDV model requires training NMT models, it is trained on the same standard parallel data as in the previous studies. This is different from adopting an external translator (e.g., Google translator), which would bring in extra information and makes the comparison unfair. Moreover, the decoding time of the translator is also faster than parameter training during testing. Nevertheless, we would like to mention that the $para\_doc$ model has a simpler structure and would train faster than ours.

Although different DV and SV variants may be produced by our NTDV model, we are glad to see that the performances of DV/SV derived from only one language (e.g., DV$_{en}$ or SV$_{en}$) are not very far behind those DV/SV derived from two languages (e.g., DV$_{en:de}$ or SV$_{en:de}$). Firstly, since they are produced by half of the NTDV model, they can be produced faster. Secondly, it is easy to expand the cross-lingual framework into a multi-lingual framework. For instance, in a plausible multilingual NTDV modeling framework involving n languages, one language (e.g., English) may be designated as the pivot language, and an NMT model is separately trained between the pivot language and each of the remaining $(n-1)$ languages to obtain NMT$_{en\rightarrow b}$, NMT$_{en\rightarrow c}$, NMT$_{en\rightarrow d}$, etc. where b, c, d represent different languages. Then an NTDV model may be constructed with all the n NMT models together with the two shared layers similar to the structure shown in Figure 1(a).

Table 2: *The classification accuracy on the CLDC task (%).*

Method	en→de	de→en	Training Corpora
$para\_doc$ * [14]	92.7	**91.5**	parallel
BAE [7]	91.8	74.2	parallel + mono
UnsupAlign [26]	90.7	80.0	parallel
BRAVE [8]	89.7	80.1	parallel
MultiVec [9]	88.2	79.1	parallel
ADD [6]	86.4	74.7	parallel
BI [6]	86.1	79.0	parallel
MT_base [12]	68.1	67.4	-
DV$_{en}$	92.58	82.53	parallel
DV$_{de}$	92.16	79.38	parallel
DV$_{en:de}$	**94.39**	82.70	parallel
DV$_{en+de}$	93.52	82.25	parallel
SV$_{en}$	93.30	82.55	parallel
SV$_{de}$	91.74	79.55	parallel
SV$_{en:de}$	94.30	83.18	parallel
SV$_{en+de}$	93.22	81.37	parallel

5. Conclusion

In this paper, we present a NTDV model to produce cross-lingual document embeddings. The NTDV model can produce good cross-lingual document vectors fast in a forward-pass of the model. Thus, it would be especially useful in systems that have a stringent limitation in classification time (such as in an online system).

6. Acknowledgements

The work described in this paper was partially supported by a grant from the Research Grants Council of the Hong Kong Special Administrative Region, China (Project No. HKUST16215816).

7. References

[1] Q. Le and T. Mikolov, "Distributed representations of sentences and documents," in *International Conference on Machine Learning*, 2014, pp. 1188–1196.

[2] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in neural information processing systems*, 2013, pp. 3111–3119.

[3] A. M. Dai, C. Olah, and Q. V. Le, "Document embedding with paragraph vectors," in *NIPS Deep Learning Workshop*, 2015.

[4] Q. Ai, L. Yang, J. Guo, and W. B. Croft, "Analysis of the paragraph vector model for information retrieval," in *Proceedings of the 2016 ACM on International Conference on the Theory of Information Retrieval*. ACM, 2016, pp. 133–142.

[5] R. Kiros, Y. Zhu, R. R. Salakhutdinov, R. Zemel, R. Urtasun, A. Torralba, and S. Fidler, "Skip-thought vectors," in *Advances in Neural Information Processing Systems 28*, C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, Eds. Curran Associates, Inc., 2015, pp. 3294–3302. [Online]. Available: http://papers.nips.cc/paper/5950-skip-thought-vectors.pdf

[6] K. M. Hermann and P. Blunsom, "Multilingual models for compositional distributional semantics," in *Proceedings of ACL*, 2014.

[7] S. C. AP, S. Lauly, H. Larochelle, M. Khapra, B. Ravindran, V. C. Raykar, and A. Saha, "An autoencoder approach to learning bilingual word representations," in *Advances in Neural Information Processing Systems*, 2014, pp. 1853–1861.

[8] A. Mogadala and A. Rettinger, "Bilingual word embeddings from parallel and non-parallel corpora for cross-language text classification," in *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2016, pp. 692–702.

[9] A. Bérard, C. Servan, O. Pietquin, and L. Besacier, "Multivec: a multilingual and multilevel representation learning toolkit for NLP," in *The 10th edition of the Language Resources and Evaluation Conference (LREC)*, 2016.

[10] D. C. Ferreira, A. F. Martins, and M. S. Almeida, "Jointly learning to embed and predict with multiple languages," in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, vol. 1, 2016, pp. 2019–2028.

[11] S. Gouws, Y. Bengio, and G. Corrado, "Bilbowa: Fast bilingual distributed representations without word alignments," in *International Conference on Machine Learning*, 2015, pp. 748–756.

[12] A. Klementiev, I. Titov, and B. Bhattarai, "Inducing crosslingual distributed representations of words," *Proceedings of COLING 2012*, pp. 1459–1474, 2012.

[13] J. Coulmance, J.-M. Marty, G. Wenzek, and A. Benhalloum, "Trans-gram, fast cross-lingual word-embeddings," in *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, 2015, pp. 1109–1113.

[14] H. Pham, T. Luong, and C. Manning, "Learning distributed representations for multilingual text sequences," in *Proceedings of the 1st Workshop on Vector Space Modeling for Natural Language Processing*, 2015, pp. 88–94.

[15] M. Amini, N. Usunier, and C. Goutte, "Learning from multiple partially observed views-an application to multilingual text categorization," in *Advances in neural information processing systems*, 2009, pp. 28–36.

[16] R. Sennrich, O. Firat, K. Cho, A. Birch, B. Haddow, J. Hitschler, M. Junczys-Dowmunt, S. Läubli, A. V. Miceli Barone, J. Mokry, and M. Nadejde, "Nematus: a toolkit for neural machine translation," in *Proceedings of the Software Demonstrations of the 15th Conference of the European Chapter of the Association for Computational Linguistics*. Valencia, Spain: Association for Computational Linguistics, April 2017, pp. 65–68. [Online]. Available: http://aclweb.org/anthology/E17-3017

[17] Z. Tu, Y. Liu, Z. Lu, X. Liu, and H. Li, "Context gates for neural machine translation," *Transactions of the Association of Computational Linguistics*, vol. 5, no. 1, pp. 87–99, 2017.

[18] Y. Ding, Y. Liu, H. Luan, and M. Sun, "Visualizing and understanding neural machine translation," in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, vol. 1, 2017, pp. 1150–1159.

[19] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems*, 2017, pp. 6000–6010.

[20] Z. Lin, M. Feng, C. N. dos Santos, M. Yu, B. Xiang, B. Zhou, and Y. Bengio, "A structured self-attentive sentence embedding," in *International Conference on Learning Representations 2017*, 2017.

[21] P. Koehn, "Europarl: A parallel corpus for statistical machine translation," in *MT summit*, vol. 5, 2005, pp. 79–86.

[22] P. Gage, "A new algorithm for data compression," *C Users J.*, vol. 12, no. 2, pp. 23–38, Feb. 1994. [Online]. Available: http://dl.acm.org/citation.cfm?id=177910.177914

[23] R. Sennrich, B. Haddow, and A. Birch, "Edinburgh neural machine translation systems for WMT 16," in *Proceedings of the First Conference on Machine Translation: Volume 2, Shared Task Papers*, vol. 2, 2016, pp. 371–376.

[24] D. Kinga and J. B. Adam, "A method for stochastic optimization," in *International Conference on Learning Representations (ICLR)*, 2015.

[25] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[26] T. Luong, H. Pham, and C. D. Manning, "Bilingual word representations with monolingual quality in mind," in *Proceedings of the 1st Workshop on Vector Space Modeling for Natural Language Processing*, 2015, pp. 151–159.