Graph based Translation Memory for Neural Machine Translation

Mengzhou Xia[*]
Carnegie Mellon University
mengzhox@andrew.cmu.edu

Guoping Huang
Tencent AI Lab
donkeyhuang@tencent.com

Lemao Liu
Tencent AI Lab
redmondliu@tencent.com

Shuming Shi
Tencent AI Lab
shumingshi@tencent.com

Abstract

A translation memory (TM) is proved to be helpful to improve neural machine translation (NMT). Existing approaches either pursue the decoding efficiency by merely accessing local information in a TM or encode the global information in a TM yet sacrificing efficiency due to redundancy. We propose an efficient approach to making use of the global information in a TM. The key idea is to pack a redundant TM into a compact graph and perform additional attention mechanisms over the packed graph for integrating the TM representation into the decoding network. We implement the model by extending the state-of-the-art NMT, Transformer. Extensive experiments on three language pairs show that the proposed approach is efficient in terms of running time and space occupation, and particularly it outperforms multiple strong baselines in terms of BLEU scores.

Introduction

A translation memory (TM) typically consists of bilingual sentence pairs that are most similar to the sentence to be translated (Robinson 2012). In computer-aided translation, professional translators are able to quickly figure out the faithful translation by standing on top of a TM instead of translating from scratch. In statistical machine translation (SMT), various research work have also been devoted to making use of a TM to improve translation quality (Simard and Isabelle 2009; Koehn and Senellart 2010; Ma et al. 2011).

Recent years have witnessed an evolutionary shift from SMT to neural machine translation (NMT), and many notable works investigate on how to integrate a TM into neural translation models (Li, Zhang, and Zong 2016; Zhang et al. 2018; Gu et al. 2018). For example, Zhang et al. (2018) propose a simple approach which defines a quantity over a TM to bias word selection for NMT decoding. The quantity is calculated based on n-gram matching between a TM and the sentence to be translated. This method indeed captures helpful local patterns from a TM while inescapably omitting significant global information such as long distance reordering in a TM. In addition, it introduces an additional hyperparameter to balance the contribution of a TM and the NMT model. The sensitive hyperparameter is required to be tuned manually for each specific task by selecting a suitable development set. On the other hand, Gu et al. (2018) propose an alternative approach to making full use of the global information in a TM. The method encodes all bilingual sentences in a TM into a key-value memory, in which each target word and its corresponding context vector represent a pair of value and key. Nevertheless, the integration of the memory leads to considerable inefficiency in terms of both computational time and space occupation due to redundant words in both source and target side of a TM.

In this paper, we propose a new approach to integrating a translation memory into NMT which utilizes *global* information within a TM in an *efficient* fashion. The key to the proposed approach is to represent a TM using a compact structure. Specifically, our approach retrieves a translation memory consisting of multiple bilingual sentences from our training corpus. As the source sentences in a TM are similar to the input sentence, we directly ignore the source side of a TM and only focus on representing the target side. To avoid redundancy in the target side of a TM, we further pack the target side into a directed graph, where each target sentence in a TM indicates a path from the beginning to the end. Then we incorporate this packed graph into the decoding network by performing a self-attention mechanism over the graph inspired by Veličković et al. (2018).

Our approach is more efficient while keeping global information because it performs an attention mechanism over fewer attentive nodes in the packed graph, compared with the large key-value memory in Gu et al. (2018). Furthermore, our approach is more robust in practice because it does not introduce additional sensitive hyperparameters to be manually tuned like Zhang et al. (2018).

To validate the effectiveness of the proposed approach, we implement our idea on top of the state-of-the-art model Transformer (Vaswani et al. 2017). Extensive experiments on six translation tasks demonstrate the following advantages of the proposed approach:

- It is more efficient than the approach in Gu et al. (2018) in terms of both running time and space occupation;
- It delivers BLEU improvements over strong baselines in-

[*] Work was done when Mengzhou Xia was interning at Tencent AI Lab.

cluding Transformer and both approaches in Zhang et al. (2018) and Gu et al. (2018).

Preliminary

Suppose $\mathbf{x} = \langle x_1, \ldots, x_{|\mathbf{x}|} \rangle$ is a source sentence with length $|\mathbf{x}|$ and $\mathbf{y} = \langle y_1, \ldots, y_{|\mathbf{y}|} \rangle$ is the corresponding target sentence of $\mathbf{x}$ with length $|\mathbf{y}|$. Generally, for a given $\mathbf{x}$, the neural machine translation baseline, Transformer, aims to generate a translation $\mathbf{y}$ according to the conditional probability $P(\mathbf{y} \mid \mathbf{x})$ defined by neural networks:

$$P(\mathbf{y} \mid \mathbf{x}) = \prod_{i=1}^{|\mathbf{y}|} P(y_i \mid \mathbf{y}_{<i}, \mathbf{x}), \quad (1)$$

where $\mathbf{y}_{<i} = \langle y_1, \ldots, y_{i-1} \rangle$ denotes a prefix of $\mathbf{y}$ with length $i-1$. To expand each factor $P(y_i \mid \mathbf{y}_{<i}, \mathbf{x})$ in Eq.(1), Transformer adopts the encoder-decoder framework similar to the standard sequence-to-sequence learning in (Bahdanau, Cho, and Bengio 2014).

In encoding $\mathbf{x}$, an encoder employs L layers of neural networks, each layer consisting of different sub-layers, i.e. multi-head attention, residual connection, layer normalization, and feed-forward network as mentioned in (Vaswani et al. 2017). The output of the l_{th} layer is obtained from the output of the $(l-1)_{th}$ layer via four sub-layers as follows:

$$h_j^{\mathrm{E},l} = RL \circ F \circ RL \circ MH(h_j^{\mathrm{E},l-1}, \mathbf{h}^{\mathrm{E},l-1}) \quad (2)$$

where $h_j^{\mathrm{E},l}$ indicates the j_{th} hidden unit with dimension d after passing l layers during the encoding phase and particularly $h_j^{\mathrm{E},0}$ denotes the word embedding plus positional encoding of x_j; $\mathbf{h}^{\mathrm{E},l} = \langle h_1^{\mathrm{E},l}, \cdots, h_n^{\mathrm{E},l} \rangle$ denotes an $(|\mathbf{x}|, d)$-matrix, i.e. a sequence of hidden units with dimension of d; $\circ$ denotes the composition operator between two functions; MH, RL and F respectively denote the functions corresponding to the sub-layers of multi-head attention, residual connection plus layer normalization, and feed-forward network with parameters omitted for easier understanding.

Note that RL and F maps a vector to another vector and we refer enthusiastic readers to Vaswani et al. (2017) for detailed definitions. $MH(h_j, \mathbf{h})$ maps a vector h_j and a matrix $\mathbf{h}$ into another vector as follows:

$$MH(h_j, \mathbf{h}) = \varphi(\mathrm{head}_1, \cdots, \mathrm{head}_K) \quad (3)$$

where φ is a linear projection function, and

$$\mathrm{head}_k = \mathrm{softmax}\left(\frac{h_j W_k^1 \left(\mathbf{h} W_k^2\right)^\top}{\sqrt{d}}\right) \mathbf{h} W_k^3 \quad (4)$$

where W_k^1, W_k^2 and W_k^3 are transformation matrices for head k and d is the dimension of vector h_j.

During decoding phase, Transformer similarly employs L layers of neural networks, yet each layer consisting of six sub-layers as follows:

$$h_i^{\mathrm{D},l} = RL \circ F \circ RL \circ MH\big(RL \circ MH(h_i^{\mathrm{D},l-1}, \mathbf{h}_{<i+1}^{\mathrm{D},l-1}), \mathbf{h}^{\mathrm{E},L}\big) \quad (5)$$

where $h_i^{\mathrm{D},l}$ indicates the i_{th} hidden unit at l_{th} layer in the decoding phase, and in particular $h_i^{\mathrm{D},0}$ denotes the word embedding of y_{i-1} plus its positional encoding for the first layer; all other functions are defined as in Eq.(2). Note that Eq.(5) differs from Eq.(2) in an additional multi-head attention mechanism over the hidden unit sequence $\mathbf{h}^{\mathrm{E},L}$ derived from the encoding phase.

Finally, the factor $P(y_i \mid \mathbf{y}_{<i}, \mathbf{x})$ in Eq.(1) is defined as follows:

$$P(y_i \mid \mathbf{y}_{<i}, \mathbf{x}) = \mathrm{softmax}\left(\phi(h_i^{\mathrm{D},L})\right) \quad (6)$$

where ϕ is a linear network to project the hidden unit to a vector with dimension of the target vocabulary size.

Graph Representation of TM

For a source sentence $\mathbf{x}$ to be translated, we firstly use the off-the-shelf search engine Lucene to retrieve a set of sentences $\{\mathbf{x}^i | i \in [1, M]\}$ together with its translations $\{\mathbf{y}^i | i \in [1, M]\}$. Following Gu et al. (2018) and Zhang et al. (2018), we prune the retrieved set to an N-best list ($M \gg N$) by the fine-grained similarity score based on the edit-distance:

$$1 - \frac{\mathrm{dist}(\mathbf{x}, \mathbf{x}^i)}{\max(|\mathbf{x}|, |\mathbf{x}^i|)} \quad (7)$$

where dist denotes the edit-distance. This N-best list is referred to as a translation memory in this paper, which is represented by $\{(\mathbf{x}^i, \mathbf{y}^i) \mid i \in [1, N]\}$.

Gu et al. (2018) encode each word in both the source and target sides of TM as a neural memory, and repeatedly access the memory during decoding. Unfortunately, since a word even a phrase may appear repeatedly in both source and target side of a TM, redundant words are encoded multiple times, leading to a large key-value memory, where the key value pairs are determined by target words. This large memory makes the attention computation inefficient in speed, and also consumes a considerable GPU memory, which is expensive for machines in practice.

To address these issues, we propose an effective approach to representing a TM in a compact way by the following two steps in this paper.

Ignoring source side

It is observed that most source words in the TM also appear in the input sentence $\mathbf{x}$ and have already been represented by the encoder. In addition, we believe that those words in the TM yet beyond $\mathbf{x}$ may not be informative to translate the sentence $\mathbf{x}$ itself. Therefore, in our proposed model, we directly ignore the source sentences of the TM and only represent the target side. In other words, we expect the target side of the TM to directly capture relevant parts in the sentence $\mathbf{x}$ without bothering to access its source side counterpart. Since the GPU memory consumed by TM encoding is proportional to the total number of words in a TM, this trick is able to reduce memory consumption greatly. Note that this trick can only improve the computational speed to a limited extent, because it still maintains the same number of target words corresponding to key value pairs.

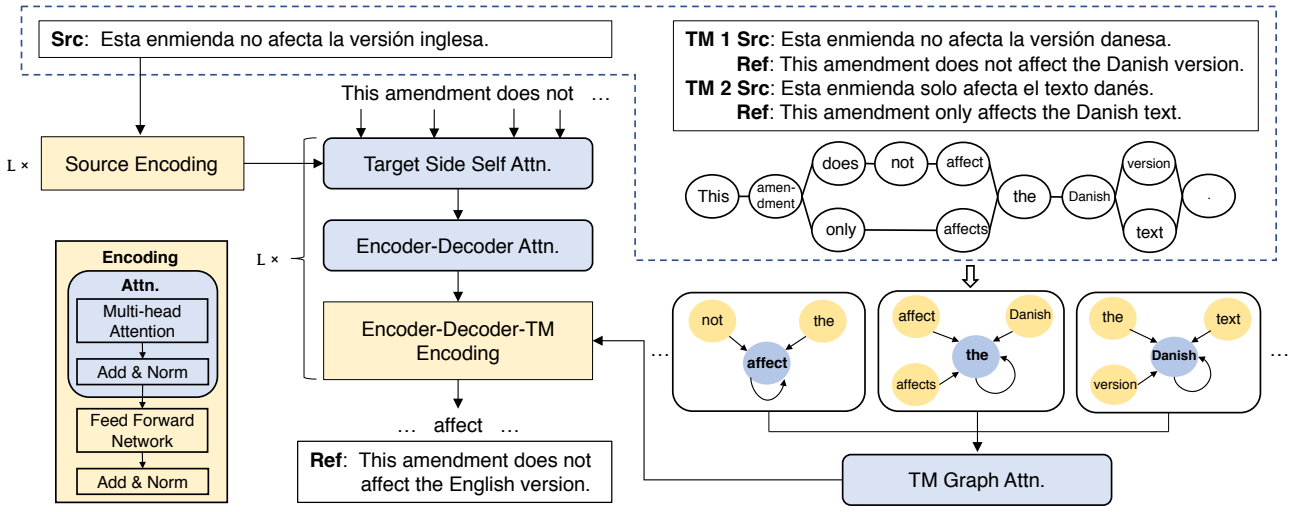

Figure 1: The architecture of the proposed NMT with graph based TM. 1) Graph representation - The part in the dashed box is a concrete example of the graph representation of a TM. The source and target languages are Spanish and English respectively. Src is the source sentence and Ref is the corresponding target sentence. Note that only 2 TM pairs are used in this example for simplicity. 2) Model architecture - The part outside of the dashed box shows the core components of the model architecture. A light blue box consists of a multi-head attention mechanism and a residual connection plus a layer normalization. A light yellow box consists of one more free-forward network and a residual connection plus a layer normalization. Specifically, the graph attention operation is presented with three selected nodes. L is the number of repetition of each part.

Packing target side

Secondly, instead of sequentially encoding target sentences in a TM, we pack them into a compact graph such that some words in different sentences may correspond to the same node in the graph, which is inspired by the notion of lattice or hypergraph in statistical machine translation (Koehn 2009). To this end, we convert the target side in a TM into a confusion network by using the algorithm proposed by Mangu, Brill, and Stolcke (1999) and Mangu, Brill, and Stolcke (2000). The basic idea is to cluster each word into an equivalent class, and organize all the classes according to an order defined over classes. Finally, we can obtain a graph via computing the dual of the confusion network, by treating edges as nodes (West and others 2001).

Figure 1 shows an example of a resulting graph for a TM (including two sentences for simplicity), where each node represents a word. In the graph, the nodes representing the words "This", "amendment", "the", "Danish", and "." appear in both sentences, but only appear once in the graph. In terms of both space and speed, it is more efficient to encode the graph, compared with the representation of all source and target words of a TM in Gu et al. (2018).

NMT with Graph based TM

Suppose $G = (\mathbf{v}, \mathbf{e})$ is the directed graph obtained from a translation memory, where $\mathbf{v} = \langle v_1, \cdots, v_{|\mathbf{v}|} \rangle$ is a sequence of nodes, and $\mathbf{e} = \langle e_1, \cdots, e_{|\mathbf{e}|} \rangle$ is a sequence of edges. In addition, $\mathbf{n}^i = \langle n_1^i, \cdots, n_{|\mathbf{n}^i|}^i \rangle$ denotes a sequence of first-order neighborhood nodes of v_i in graph G including v_i itself. Nodes are arranged in topological order. For example, as illustrated in Figure 1, $\mathbf{v}_1 = This$, $\mathbf{n}^1 = \langle This, amendment \rangle$ and $\mathbf{v}_2 = amendment$, $\mathbf{n}^2 = \langle This, amendment, does, only \rangle$.

Generally, the enhanced Transformer shares the similar architecture as the baseline but with two major differences in encoding and decoding phases. The model structure is illustrated in figure 1.

Enhanced Graph Encoding

Besides encoding the input sequence $\mathbf{x}$, the proposed model also encodes $\mathbf{v}$ from G by using L layers of networks in a similar fashion to the encoding of the input $\mathbf{x}$. Suppose $h_i^{\mathrm{G},l}$ denotes the hidden units at l_{th} layer regarding the node v_i. Inspired by the graph attention in Veličković et al. (2018), $h_i^{\mathrm{G},l}$ is calculated by the hidden units at $l-1_{th}$ layer as follows:

$$h_i^{\mathrm{G},l} = RL \circ F \circ RL \circ MH(h_i^{\mathrm{G},l-1}, \mathbf{h}^{\mathrm{G},l-1}[\mathbf{n}^i]) \quad (8)$$

where RL, F and MH are functions corresponding to sub-layers of residual connection plus layer normalization, feed-forward network and multi-head attention mechanism defined in Eq.(2). $\mathbf{h}^{\mathrm{G},l-1}[\mathbf{n}^i]$ is a matrix obtained by taking the slice over the matrix $\mathbf{h}^{g,l-1}$ along the first axis via $\mathbf{n}^i$.

Based on the encoding schema elaborated above, there are different variations of the graph encoding in real practice. We can choose to fix the graph encoding after L layers' computation and introduce the fixed encoding to each decoding layer, which is the same as the encoding of the source side $\mathbf{x}$. We can also introduce a flexible encoding to each decoding layer, which is calculated as follows:

$$h_i^{\mathrm{G},l} = RL \circ MH_l(h_i^{\mathrm{G},0}, \mathbf{h}^{\mathrm{G},0}[\mathbf{n}^i]) \quad (9)$$

where $\mathbf{h}^{\mathrm{G},0}$ denotes the word embedding plus positional encoding of the translation memory sequence. When flexible

encoding is applied, each decoding layer gets its own unique graph encoding. Besides, the outermost components RL and F are removed from flexible encoding to reduce complexity.

Because the directed graph G roughly records the global order of nodes, which somehow captures the reordering in the original TM, it would be helpful to encode the directional information for translating the input $\mathbf{x}$. In this paper, we index the nodes $\mathbf{v}$ in topological order and integrate additional positional encoding signal before the graph attention similar to the encoding of $\mathbf{x}$ as in the standard Transformer architecture. In detail, the first layer of graph encoding is the sum of the word embedding of v_i and position encoding of i (Vaswani et al. 2017).

Enhanced Decoding

Suppose $\mathbf{h}^{\mathrm{E},L}$ is the hidden unit sequence encoded from $\mathbf{x}$ and $\mathbf{h}^{\mathrm{G},L}$ is that encoded from G. As metioned in the last subsection, $\mathbf{h}^{\mathrm{G},L}$ could be a fixed graph encoding or a flexible one. Similar to Transformer, the proposed model employs L layers of networks in the decoding phase, but each layer includes two additional sub-layers (MH and RL) to incorporate $\mathbf{h}^{\mathrm{G},L}$, besides other six sub-layers. We place the extra two layers nearest to the output of the decoder network in order to let the graph encoding fully influence the decoding process. Formally, at the l_{th} layer in the decoding phase, the hidden unit $h_i^{\mathrm{D},l}$ is computed from $\mathbf{h}_i^{\mathrm{D},l-1}$, $\mathbf{h}^{\mathrm{E},L}$ and $\mathbf{h}^{\mathrm{G},L}$ as follows:

$$h_i^{\mathrm{D},l} = RL \circ F \circ RL \circ MH\Big(RL \circ MH\Big(RL \circ MH(h_i^{\mathrm{D},l-1}, \mathbf{h}_{<i+1}^{\mathrm{D},l-1}), \mathbf{h}^{\mathrm{E},L}\Big), \mathbf{h}^{\mathrm{G},L}\Big) \quad (10)$$

This equation Eq.(10) is similar to Eq.(5) except two extra sub-layers including MH over $\mathbf{h}^{\mathrm{G},L}$ and its immediate RL. By applying these additional layers, the model is able to dynamically extract favorable semantic and syntactic information from translation memory for each time step.

Note that in encoding v_i, the sequence of $\mathbf{n}^i$ only includes the nodes which are directly adjacent to v_i, one might argue that the hidden unit $h_i^{\mathrm{G},l}$ ignores information far away from the node v_i. Indeed, we tried to reset $\mathbf{n}^i$ to include higher order neighbors of v_i, but we did not observe additional gains in translation quality. The possible reason is that the long distance information is captured by the multi-layer structure.

Related Work

This section reviews the mostly related works, according to the following three research lines.

TM based SMT

Many research works are devoted to integrating a translation memory into the statistical machine translation paradigm. For example, Koehn and Senellart (2010) extract bilingual segments from a TM which matches the source sentence to be translated, and constrain SMT to decode for those unmatched parts of the source sentence. Unlike Koehn and Senellart (2010) employing a heuristic score to decide whether the extracted segments should be used as decoding constraints or not, Ma et al. (2011) design a fine-grained classifier to predict the score for making more reliable decisions. In addition, Simard and Isabelle (2009), Wang, Zong, and Su (2013) and Wang, Zong, and Su (2014) add the extracted bilingual segments to the translation table of SMT, and then decode the source sentence with the augmented translation table, which bias the decoder in a soft constraint manner instead of a hard one as in Koehn and Senellart (2010) and Ma et al. (2011).

TM based NMT

Recently, TM based neural machine translation has been witnessed increasing interests. As NMT does not explicitly rely on the translation rules as SMT, many works resort to different approaches. For example, Li, Zhang, and Zong (2016) and Farajian et al. (2017) make use of a translation memory to fine tune the NMT model which is pre-trained on the entire training corpus in advance, similar to that in Liu et al. (2012) in SMT. As different testing sentences may have different TMs, this approach has to fine-tune the model on-the-fly during testing. In order to avoid the on-the-fly tuning, Zhang et al. (2018) define a quantity based on the n-gram in TM to directly bias the word selection for NMT. Although this approach is efficient, it can only captures local information in a hard manner while ignoring the global information in TM, leading to inferior performance to ours. Gu et al. (2018) propose an additional encoder to encode the global information (i.e. the entire sentences) in TM into vectors for all words, and make use of the encoded vectors to decode a target word by an additional attention mechanism. To some extent, our approach is an extension of Gu et al. (2018) in which we pack the sequential TM into a graph, which includes much fewer nodes, leading to a more efficient attention computation.

Graph based NMT

There are many works involving graph structures to improve NMT. For instance, Stahlberg et al. (2016) implicitly employs a translation graph (i.e. lattice) generated by SMT by rewarding the words which have high posterior probabilities over the lattice. Meanwhile, Khayrallah et al. (2017) explicitly use a lattice generated by SMT to constrain the decoding space and rerank to select the best translation from this constrained lattice. Both approaches obviously differ from ours in that they do not encode the discrete graph into the continuous vectors from the point view of neural networks. Ma et al. (2018) and Su et al. (2017) encode a packed parse forest or a word segmentation lattice of a source sentence. However, the focus of our work is oriented to a translation memory instead of a parse tree or word segmentation. Furthermore, we integrate the encoded vectors into the strong Transformer, which leads to nontrivial implementations compared with attention operations in recurrent networks.

Experiments

In this section, we demonstrate, by experiments, the advantages of the proposed model: it yields better translation than

Zhang et al. (2018) with the help of global information from translation memory; and it is more efficient than the model in Gu et al. (2018) in terms of running time and memory consumption mainly because of the compact graph representation of translation memory.

Settings

Data Preparation Following the previous works investigating on incorporating TM into NMT models, we use the JRC-Acquis corpus for training and evaluating our proposed model. The JRC-Acquis corpus is a collection of parallel legislative text of European union Law applicable in the EU member states. The highly related text in the corpus is suitable for us to make evaluations. To fully explore the effectiveness of our proposed model, we conduct translation experiments on three language pair bidirectionally, namely, en-fr, en-es, and en-de.

We manage to obtain preprocessed datasets from Gu et al. (2018). For each language pair, we randomly select 3000 samples to form a development and a test set respectively. The rest of the pairs are used as the training set. Sentences longer than 80 and 100 are removed from the training and development/test set. The technique of Byte-pair Encoding (Sennrich, Haddow, and Birch 2016) is applied and the vocabulary size is set to be 20K for all the experiments.

Baseline Systems The proposed TM graph based model is built on transformer (Vaswani et al. 2017), and it is denoted by **G-TFM**. We also propose and implement another simpler TM representation based transformer, **SEQ-TFM** which sequentially encodes all target sentences in a TM as one of the baseline models. Specifically, each target sentence in TM goes through a multi-head mechanism and an immediate residual connection plus layer normalization in l_{th} layer, which is calculated as follows:

$$h^l_{k,i} = RL \circ MH_l(h^0_{k,i}, \mathbf{h^0_k}) \tag{11}$$

where $h^l_{k,i}$ is the i_{th} hidden unit of the k_{th} sentence in the TM; $h^0_{k,i}$ is the word embedding plus positional encoding of k_{th} sentence in TM. The derived representations for these sentences are then concatenated to form the representation of the translation memory $h^{\mathrm{S},l}$, which can be utilized flexibly in l_{th} decoding layer.

Besides, as the proposed model is directly built upon the Transformer architecture, Transformer itself provides a natural baseline, which is referred to as **TFM** in this paper. In addition, following Zhang et al. (2018) and Gu et al. (2018), we implement two translation memory based systems on top of Transformer for fair comparison and we refer them to **P-TFM** and **SEG-TFM**, respectively. [1] For the broader comparison, we reproduce the model (denoted by **P-RNN**) in Zhang et al. (2018) on top of our in-house RNNsearch system (denoted by **RNN**). [2]

Word embedding	512
Layers	6
TM dropout	0.6
Other dropout	0.1
Beam size	5
Label smoothing	0.1
Batch size (tokens)	8192

Table 1: Hyper-parameters for training baselines and the proposed model on top of the transformer architecture.

Training systems For each sentence, we retrieve 100 translation pairs from the training set by using Apache Lucene. We score the source side of each retrieved pair against the source sentence x with fuzzy matching score in Eq. (7) and select top $N = 5$ translation sentence pairs as a translation memory for the sentence x to be translated. Sentences from the target side in the translation memory are used to form a graph, with each word represented as a node and the connection between adjacent words in a sentence represented as an undirected edge.

For training all systems, we maintain the same hyperparameters as shown in Table 1 for comparison. Besides, we adopt the same training algorithm to learn the models as follows. We use a customized leaning rate decay paradigm following Tensor2Tensor(Vaswani et al. 2018) package. The learning rate increases linearly on early stages for a certain number of steps, known as warm-up steps, and decay exponentially later on. We set the warm-up step to be 5 epochs and we early stop the model after training 20 epochs, typically the time when the development performance varies insignificantly.

Furthermore, since there is a hyperparameter in the system P-TFM of Zhang et al. (2018) which is sensitive to the specific translation task, we tune it carefully on the development set for all translation tasks. Its optimized value is 0.7 for es and de tasks while it is 0.8 for fr task. [3]

Results and analysis on es-en task

Translation accuracy Table 2 shows the experiment results of all the systems on the es-en task in terms of BLEU. Several observations can be made from the results. First, the baseline TFM achieves substantial gains over RNN and even outperforms P-RNN by around 1 BLEU point on the test set. Compared with the strongest baseline P-TFM, the proposed SEQ-TFM and G-TFM are able to obtain some gains up to 1.9 BLEU points on the test set. This result verifies that our

[1] Note that due to the architecture divergence between RNN-based NMT and Transformer, SEG-TFG differs from the RNN-based counterpart in that two quantities c_t and z_t in Gu et al. (2018) are replaced by the hidden units obtained from the multi-head attention over the encoding units and the decoding hidden state units before the softmax operator.

[2] We did not reimplement the model in Gu et al. (2018) on top of our in-house RNN, because it was clearly demonstrated in Zhang et al. (2018) that P-RNN works comparable to or even better than Gu et al. (2018) on the same JRC-Acquis corpora as conducted in our experiments.

[3] We run all 6 tasks with hyperparameters among [0.5, 1.5] with scale of 0.1, and manually pick the optimized value according to its performance on the development set.

	RNN	P-RNN	TFM	P-TFM	SEG-TFM	SEQ-TFM	G-TFM
Dev	57.74	60.87	62.78	63.97	63.16	64.81	**66.37**
Test	58.06	61.52	62.68	64.30	62.94	65.16	**66.21**

Table 2: Translation accuracy in terms of BLEU on the es-en task.

	TFM	SEG-TFM	SEQ-TFM	G-TFM
Train (s)	4579	44238	21920	8692
Test (s)	0.20	2.68	1.25	0.36
Words (#)	68.28	374.52	214.97	129.18
BLEU	62.68	62.94	65.16	**66.21**

Table 3: Running time and memory. Training time reports the time in seconds for training one epoch on average, and testing time reports the time in seconds for translating one sentence on average. Words (#) denotes the number of words encoded in the neural models on average.

compact representation of TM is able to guide the decoding of the state-of-the-art model.

Second, it is observed that SEG-TFM is only comparable to TFM on this task, although its RNN based counterpart brought significant gains as reported in Gu et al. (2018). This fact shows that the transformer architecture may need a sophisticated way to well define a key-value memory for TM encoding, which can be significantly different from that on RNN architecture. This is beyond the scope of this paper. Fortunately, this paper provides an easy yet effective approach to encode a TM, i.e. G-TFM, which does not rely on a context-based key-value memory.

Running time Since the retrieval time can be neglected compared with the decoding time as found in Zhang et al. (2018), we thereby eliminate the retrieval time and directly compare running time for neural models as shown in Table 3. From this table, we observe that the proposed graph based model G-TFM saves significant running time compared with SEG-TFM and SEQ-TFM while achieving better translation performance.

Memory consumption Ideally, the total memory consumed by a system is proportional to the size of the computational tensor graph. However, it is impossible to exactly compare the complexity of the computational graph, because it is dependent on the coding implementation. [4] As a result, we roughly evaluate the memory consumption in terms of the number of words encoded by the models, which corresponds to a part of the nodes in the computational graph and can be calculated.

Table 3 depicts the total number of source and target words encoded by the corresponding model for each test sentence on average. It's observed that SEG-TFM needs to encode approximately 3 times and SEQ-TFM encodes approximately 2 times the number of words of our proposed model, G-TFM. There's no wonder that TFM takes the fewest words to encode because no extra TM is included. These statistics indicate that under the scenario of incorporating TM in NMT, our model requires the least memory.

[4]Different implementation may lead to different computational graphs due to temporary variables.

Similarity	Sents	Percent	TFM	P-TFM	G-TFM
$[0, 0.1)$	2	0.19%	54.17	68.43	58.22
$[0.1, 0.2)$	138	11.40%	36.08	36.46	37.49
$[0.2, 0.3)$	462	20.76%	44.94	45.99	45.30
$[0.3, 0.4)$	305	14.56%	51.19	51.98	51.75
$[0.4, 0.5)$	272	12.83%	60.72	62.86	62.10
$[0.5, 0.6)$	206	7.82%	66.31	66.41	70.66
$[0.6, 0.7)$	203	7.40%	71.38	73.73	76.17
$[0.7, 0.8)$	188	7.05%	77.48	78.78	83.37
$[0.8, 0.9)$	377	10.90%	83.81	85.52	88.71
$[0.9, 1)$	432	7.09%	87.81	88.02	93.21
$[0, 1)$	2585	100%	62.68	64.30	66.21

Table 4: Translation quality on es-en task for the divided subsets according to similarity. The averaged similarity for the entire test set is 0.48.

	TFM	P-TFM	G-TFM
BLEU	62.68	65.00	**90.25**

Table 5: Translation results when adding reference to TM.

Influence on similarity Intuitively, the performance of our proposed model is supposed to vary on the basis of the similarity between the sentences to be translated and the retrieved translation memory. To verify the hypothesis, we divided the test set into bucketed subsets based on the averaged similarity of each sentence in the retrieved translation memory. From the results reported in Table 4, it is observed that the proposed G-TFM outperforms the baseline TF on all bucketed subsets in general. As the similarity score increases, G-TFM leads to more significant improvements than those of the baseline TFM. In detail, the improvements over TFM are up to 5 BLEU points when similarity is around 0.9; while the improvements are less than 1 BLEU as the similarity score is between 0.2 and 0.3. This fact shows that the translation memory indeed brings improvements of translation quality over all similarity buckets. And the more similar the TM is, the better it will do.

In addition, P-TFM is slightly better than the proposed G-TFM when the similarity is between 0.2 and 0.5. But G-TFM outperforms P-TFM with a substantial margin if the similarity is more than 0.5. In particular, as the similarity is above 0.9, G-TFM delivers a gain of 5 BLEU over P-TFM.

Example	Fuzzy Matching Score: *0.31*, T1 BLEU: *0.53*, T2 BLEU: *0.56*, T3 BLEU: ***0.75***		
S	**Dentro de los lmites establecidos en el anexo III** , se conceder un suplemento del pago por superficie de 344,5 euros por hectrea por las superficies sembradas de trigo duro en las zonas tradicionales de produccin que figuran en la lista del anexo II .	**T1**	**Within the limits laid down in Annex III** , a supplement to the area payment of EUR 344,5 per hectare shall be granted for the areas sown with durum wheat in the traditional production zones listed in Annex II .
R	A supplement to the area payment of EUR 344,50 per hectare shall be paid for the area down to durum wheat in the traditional production zones listed in Annex II , **subject to the limits fixed in Annex III .**	**T2**	**Within the limits laid down in Annex III** , a supplement to the area payment of EUR 344,5 per hectare shall be granted for the areas sown to durum wheat in the traditional production zones listed in Annex II .
TM	shall be paid for the area down to durum wheat in the traditional production zones listed in Annex X , **subject to the following limits** :	**T3**	A supplement to the area payment of EUR 344,5 / ha for the areas down to durum wheat in the traditional production zones listed in Annex II , **subject to the limits fixed in Annex III** .

Table 6: An example of effects on using global information from es-en test set. **S** and **R** respectively denote source and reference sentences, **TM** shows only one sentence in the given translation memory, **T1**, **T2** and **T3** represent TFM, P-TFM and G-TFM.

	BLEU	en-fr	fr-en	en-de	de-en	en-es
Dev	TFM	66.33	65.95	53.32	58.54	60.43
	P-TFM	68.90	68.61	55.54	60.10	61.50
	G-TFM	**69.69**	**70.65**	**57.43**	**61.85**	**62.50**
Test	TFM	66.36	66.96	53.29	58.86	60.52
	P-TFM	68.73	68.70	55.14	60.26	61.56
	G-TFM	**69.59**	**70.87**	**56.88**	**61.72**	**62.76**

Table 7: Translation Results on both development and test sets across other 5 translation tasks.

One main reason is that P-TFM is highly sensitive to the value of the reward defined in Zhang et al. (2018), which is determined by the sentence similarity and the hyperparameter. To further support this, we add the reference into TM for testing to evaluate the performance of a well-trained existing model. We compare the translation results of P-TFM and G-TFM again. Table 5 shows that the gap between P-TFM and G-TFM is substantially enlarged. This result shows that our proposed G-TFM is able to adaptively make use of translation memory. A well-trained existing model can deliver almost absolutely correct translations when a extremely similar translation memory is available. But for P-TFM, the hyperparameter previously tuned on the development set is not suitable for the translation task where the TM is very similar to test sentences. In other words, the success of P-TFM also depends on the selection of a development set.

Case study on global information In the example shown in Table 6, it is clear that there is a long distance reorder between the main and subordinate clauses, and the baseline TFM fails to figure out this reorder as the reference. Since the reward score of any n-gram defined in Zhang et al. (2018) is all the same and irrelevant to its position information, P-TFM can not encourage to make a long-distance reordering decision. G-TFM succeeds to capture the long distance reordering with positional encoding in this case by learning from the similar pattern contained in the TM, which indicates that G-TFM indeed is able to utilize the global information in TM.

Results on other tasks

We pick stronger baselines from the es-en task, i.e. TFM and P-TFM, and compare them with the proposed G-TFM model on other 5 translation tasks. Table 7 summarizes their results on both the development and test sets. From this table, we can see that on the test set, G-TFM steadily outperforms TFM by up to 3 BLEU points across all these 5 tasks, In addition, contrast to P-TFM, G-TFM demonstrates better performance by exceeding at least 1 BLEU point across all these tasks except the en-fr task. These results are consistent with the results on es-en task and further validates the effectiveness of integrating graph-based translation memory into the Transformer model.

Conclusion

We have proposed an approach to augmenting the neural machine translation by a translation memory. The proposed approach firstly packs the translation memory into a compact graph, where each node may correspond to multiple words for different sentences in a TM, and then it encodes the packed graph into a deep representation during the decoding phase. The proposed approach is able to make use of the global information from TM and is also efficient enough even if the size of TM increases. Extensive experiments on six tasks with three language pairs show that the proposed approach is effective when compared with the strong baseline Transformer: it not only gains over the baseline with a large margin, but also consistently outperforms the approaches in Zhang et al. (2018) and Gu et al. (2018).

Acknowledgments

We thank Jiatao Gu and Jingyi Zhang for their help on the datasets and reviewers for their valuable feedbacks on this paper.

References

Bahdanau, D.; Cho, K.; and Bengio, Y. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Farajian, M. A.; Turchi, M.; Negri, M.; and Federico, M. 2017. Multi-domain neural machine translation through unsupervised adaptation. In *Proceedings of the Second Conference on Machine Translation*, 127–137.

Gu, J.; Wang, Y.; Cho, K.; and Li, V. O. K. 2018. Search engine guided neural machine translation. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, New Orleans, Louisiana, USA, February 2-7, 2018*.

Khayrallah, H.; Kumar, G.; Duh, K.; Post, M.; and Koehn, P. 2017. Neural lattice search for domain adaptation in machine translation. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, volume 2, 20–25.

Koehn, P., and Senellart, J. 2010. Convergence of translation memory and statistical machine translation. In *Proceedings of AMTA Workshop on MT Research and the Translation Industry*, 21–31.

Koehn, P. 2009. *Statistical machine translation*. Cambridge University Press.

Li, X.; Zhang, J.; and Zong, C. 2016. One sentence one model for neural machine translation. *arXiv preprint arXiv:1609.06490*.

Liu, L.; Cao, H.; Watanabe, T.; Zhao, T.; Yu, M.; and Zhu, C. 2012. Locally training the log-linear model for smt. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, 402–411. Association for Computational Linguistics.

Ma, Y.; He, Y.; Way, A.; and van Genabith, J. 2011. Consistent translation using discriminative learning: a translation memory-inspired approach. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, 1239–1248. Association for Computational Linguistics.

Ma, C.; Tamura, A.; Utiyama, M.; Zhao, T.; and Sumita, E. 2018. Forest-based neural machine translation. In *Proceedings of ACL*.

Mangu, L.; Brill, E.; and Stolcke, A. 1999. Finding consensus among words: Lattice-based word error minimization. In *Sixth European Conference on Speech Communication and Technology*.

Mangu, L.; Brill, E.; and Stolcke, A. 2000. Finding consensus in speech recognition: word error minimization and other applications of confusion networks. *Computer Speech & Language* 14(4):373–400.

Robinson, D. 2012. *Becoming a translator: An introduction to the theory and practice of translation*. Routledge.

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 (Volume 1: Long Papers)*, 1715–1725. Berlin, Germany: Association for Computational Linguistics.

Simard, M., and Isabelle, P. 2009. Phrase-based machine translation in a computer-assisted translation environment. *Proceedings of the Twelfth Machine Translation Summit (MT Summit XII)* 120–127.

Stahlberg, F.; Hasler, E.; Waite, A.; and Byrne, B. 2016. Syntactically guided neural machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, 299–305. Berlin, Germany: Association for Computational Linguistics.

Su, J.; Tan, Z.; Xiong, D.; Ji, R.; Shi, X.; and Liu, Y. 2017. Lattice-based recurrent neural network encoders for neural machine translation. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA.*, 3302–3308.

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

Vaswani, A.; Bengio, S.; Brevdo, E.; Chollet, F.; Gomez, A. N.; Gouws, S.; Jones, L.; Kaiser, Ł.; Kalchbrenner, N.; Parmar, N.; et al. 2018. Tensor2tensor for neural machine translation. *arXiv preprint arXiv:1803.07416*.

Veličković, P.; Cucurull, G.; Casanova, A.; Romero, A.; Liò, P.; and Bengio, Y. 2018. Graph attention networks. In *Proceedings of ICLR*.

Wang, K.; Zong, C.; and Su, K.-Y. 2013. Integrating translation memory into phrase-based machine translation during decoding. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, 11–21.

Wang, K.; Zong, C.; and Su, K.-Y. 2014. Dynamically integrating cross-domain translation memory into phrase-based machine translation during decoding. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, 398–408.

West, D. B., et al. 2001. *Introduction to graph theory*, volume 2. Prentice hall Upper Saddle River.

Zhang, J.; Utiyama, M.; Sumita, E.; Neubig, G.; and Nakamura, S. 2018. Guiding neural machine translation with retrieved translation pieces. In *Meeting of the North American Chapter of the Association for Computational Linguistics (NAACL)*.