Resolving Lexical Ambiguity in English–Japanese Neural Machine Translation

Quang Minh Do, Incheon Paik
School of Computer Science and Engineering, University of Aizu, Fukushima, Japan
`quminh.do@gmail.com, paikic@u-aizu.ac.jp`

Abstract

Lexical ambiguity, i.e., the presence of two or more meanings for a single word, is an inherent and challenging problem for machine translation systems. Even though the use of recurrent neural networks (RNN) and attention mechanisms are expected to solve this problem, machine translation systems are not always able to correctly translate lexically ambiguous sentences. In this work, we attempt to resolve the problem of lexical ambiguity in English–Japanese neural machine translation systems by combining a pretrained Bidirectional Encoder Representations from Transformer (BERT) language model that can produce contextualized word embeddings and a Transformer translation model, which is a state-of-the-art architecture for the machine translation task. These two proposed architectures have been shown to be more effective in translating ambiguous sentences than a vanilla Transformer model and the Google Translate system. Furthermore, one of the proposed models, the $\mathbf{Transformer_{BERT-WE}}$, achieves a higher BLEU score compared to the vanilla Transformer model in terms of general translation, which is concrete proof that the use of contextualized word embeddings from BERT can not only solve the problem of lexical ambiguity, but also boosts the translation quality in general.

1 Introduction

Machine translation is one of the most important tasks in the field of natural language processing. In 2014, Sutskever and his fellow researchers at Google introduced the sequence-to-sequence (seq2seq) model (Sutskever et al., 2014), marking the advent of neural machine translation (NMT) in a breakthrough in the field of machine translation. Since then, seq2seq models have been growing rapidly, evolving from a purely recurrent neural network (RNN)-based encoder–decoder model to recurrence-free models that rely on convolution(Gehring et al., 2017) or attention mechanisms(Vaswani et al., 2017). The Transformer architecture(Vaswani et al., 2017), which is based on attention mechanism, is currently the standard model for machine translation tasks because of its effectiveness and efficiency. It also provides a foundation for the advent of state-of-the-art language models, such as Bidirectional Encoder Representations from Transformer (BERT)(Devlin et al., 2018) and GPT-2(Radford et al., 2019). Section 2 shows how seq2seq models transformed from a purely RNN-based encoder–decoder model to a transformer model that relies entirely on attention mechanism.

Although many significant improvements have been made in the NMT field, lexical ambiguity is still a problem that causes difficulty for machine translation models. Liu et al. (2017)(Liu et al., 2017) show that the performance of RNN-based seq2seq model decreases as the number of senses for each word increases. Section 3 demonstrates that even modern translation models, such as Google Translate, cannot translate some lexically ambiguous sentences and forms hypotheses concerning some causes of this problem. Section 4 describes the BERT language model and explains why BERT vector representations can help resolve the problem of lexical ambiguity. Subsequently, two context-aware machine translation architectures that integrate pretrained BERT and Transformer models are proposed in section 5. For comparison purposes, a vanilla Transformer was built with the same set of hyperparameters and trained with the same settings as the proposed models. Finally, the three models were evaluated based on two criteria: i.e., the capability to produce good translations in general and the ability to translate lexically ambiguous sentences. The evaluation results and sample translations are shown in section

6.3.

2 Neural machine translation

2.1 Sequence-to-sequence model

NMT is an approach to machine translation, where a large neural network model learns to predict the likelihood of a sequence of words given a source sentence in an end-to-end fashion. The neural network model used for machine translation is called a seq2seq model, which is composed of an encoder and a decoder. RNN and its variants such as long short-term memory (LSTM) and gated recurrent unit (GRU) have been a common choice to build a seq2seq model. The encoder, which is a multilayered RNN cell, encodes the input sequence $\mathbf{x}$ into a fixed-sized vector $\mathbf{v}$, which is essentially the last hidden state of the encoder's RNN. The decoder, which is another RNN, maps this context vector to the target sequence $\mathbf{y}$. In other words, a seq2seq model learns to maximize the conditional probability:

$$\begin{aligned} &p(\mathbf{y}_1, \ldots, \mathbf{y}_T | \mathbf{x}_1, ..., \mathbf{x}_S) \\ &\qquad = \prod_{t=1}^{T} p(\mathbf{y}_t | \mathbf{v}, \mathbf{y}_1, \ldots, \mathbf{y_{t-1}}) \end{aligned} \quad (1)$$

where T and S are the lengths of the input sentence of the source language and the output sentence of the target language, respectively.

2.2 Attention-based Neural Machine Translation

The attention mechanism proposed by Bahdanau et al. (2014)(Bahdanau et al., 2014), is a significant improvement to seq2seq models. By using the attention mechanism, each position in the decoder can selectively focus on all positions in the encoder instead of relying entirely on the last hidden state of the encoder, which consequently boosts the model's capability to learn long-term dependencies.

Basically, the attention mechanism is a mapping of a query and a set of key-value pairs to an output vector. Each query vector represents a decoder's hidden state $\mathbf{h}_t$, while the key and values vectors represent all the encoder's hidden states $\mathbf{h}_s$. The output vector $\mathbf{c}_t$ is a weighted sum of the value vectors, where the weight corresponding to each value vector is computed by an alignment function.

$$\begin{aligned} \mathbf{a}_t(s) &= \text{align}(\mathbf{h}_t, \mathbf{h}_s) \\ &= \frac{\exp(\text{similarity}(\mathbf{h}_t, \mathbf{h}_s))}{\sum_{s'} \exp(\text{similarity}(\mathbf{h}_t, \mathbf{h}_{s'}))} \end{aligned} \quad (2)$$

where the value of the *similarity* function describes to what extent an input at position s and an output at position t match. The two most commonly used *similarity* functions are additive attention and multiplicative attention, which were proposed by Bahdanau et al. (2014)(Bahdanau et al., 2014) and Luong et al. (2015)(Luong et al., 2015) respectively, as shown in Eq. 3.

$$\text{similarity}(\mathbf{h}_t, \mathbf{h}_s) = \begin{cases} \mathbf{v}_a^\top \tanh(\mathbf{W}_a[\mathbf{h}_t; \mathbf{h}_s]) \\ \mathbf{h}_t^\top \mathbf{h}_s \end{cases} \quad (3)$$

2.3 Transformer

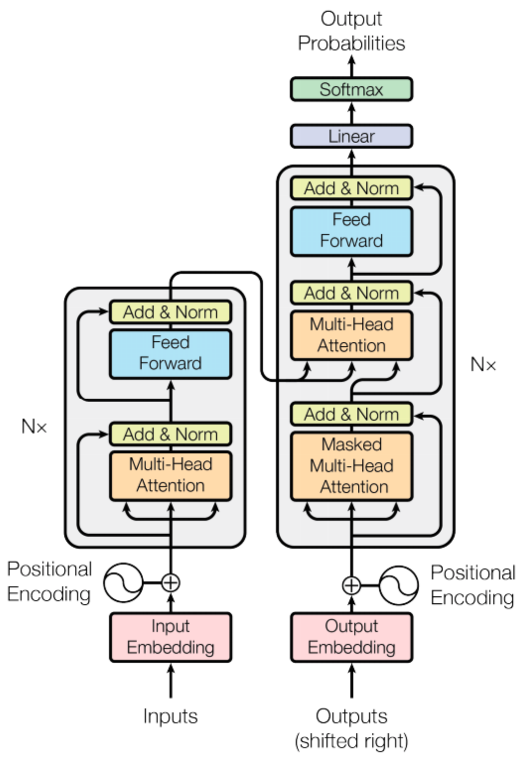

Figure 1: The Transformer architecture

The Transformer model was first introduced by Vaswani et al. (2017)(Vaswani et al., 2017), which removes all recurrence and relies entirely on self-attention. In a self-attention layer, all queries, keys, and values come from the same place. As a result, each position can attend to all other positions in the same sequence, which dispenses with the need for recurrence. This architecture does not only outperform RNN-based seq2seq models in terms of performance, but also it is parallelizable and requires less time to train. Thus, it has replaced the RNN-based seq2seq model as the de facto standard in neural machine translation.

Like other seq2seq models, a Transformer consists of an encoder and a decoder, as shown in Figure 1. The encoder is a stack of $N = 6$ identical layers, each of which is composed of two linked sublayers: a multihead attention mechanism and a fully connected feed-forward network. The decoder stack also consists of $N = 6$ identical layers. Unlike the encoder, each decoder layer is composed of three consecutive sublayers: a masked multihead attention, a multihead attention mechanism, and a fully connected feed-forward network. The first attention sublayer in each decoder layer performs self-attention, while the second one pays attention to the output of the encoder stack. In both the encoder and decoder, each sublayer's input and output are added using a residual connection and normalized using a layer normalization method (Ba et al., 2016). All sublayers in the model and the embedding layers produce outputs of dimension $d_{model} = 512$.

The Transformer model uses a multihead attention mechanism, i.e., many attention functions are performed simultaneously in the same attention layer. Specifically, all queries, keys, and values of dimension d_{model} are projected $h = 8$ times with different learned linear projections to d_k, d_k, and d_v dimensions, respectively ($d_k = d_v = d_{model}/h = 64$). Each attention head produces output vectors of dimension d_v, which are concatenated and linearly projected one more time to produce the final output vectors of dimension d_{model}.

The alignment function used in the Transformer model is called scaled dot-product attention:

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^\top}{\sqrt{d_k}}) \qquad (4)$$

Each fully connected feed-forward network in the model consists of a hidden layer with ReLU activation and an output layer, producing outputs of dimensions $d_{ff} = 2048$ and $d_{model} = 512$, respectively.

$$\text{FFN}(x) = \max(0, xW_1 + b_1)W_2 + b2$$

The self-attention mechanism uses symmetrical matrix multiplication; therefore, it cannot capture information about the sequence order. Thus, in addition to word embedding layers, it is necessary to add some information about the relative or absolute positions of the tokens in the sequence(Vaswani et al., 2017). The positional encodings are added to the outputs of embedding layers before being fed to the encoder and decoder stacks. For the model to attend to relative positions, the sine and cosine functions of different frequencies are used to generate positional encodings:

$$\text{PE}(pos, 2i) = \sin(pos/10000^{2i/d_{model}}) \qquad (5)$$

$$\text{PE}(pos, 2i+1) = \cos(pos/10000^{2i/d_{model}}) \qquad (6)$$

where pos is the position and i is the dimension.

3 English–Japanese Neural Machine Translation's Problem with Lexical Ambiguity

Lexical ambiguity, which is also called semantic ambiguity, can be defined as the presence of more than one meaning for a single word. Words that possess two or more possible meanings are called homographs. Translating homographs is not a trivial task because their meanings vary based on their contexts. For example, given a sentence "*The fisherman went to the bank.*", the word "*bank*" may refer to "*a financial institution*" or "*the side of a river*." In this case, it is acceptable to interpret this sentence in two ways. However, given another sentence "A fisherman is sitting on the bank.", it is unreasonable to interpret the word "*bank*" as "*a financial institution*" in this case. However, this sentence is challenging for machine translation systems to produce a correct translation, as shown in the later part of the paper.

Even though many advancements have been made in the field of neural machine translation to date, contemporary translation systems are still struggling to deal with semantic ambiguity. For instance, although the sentence "*He **left** a book on the table and **left** the room.*" contains two words "*left*" of different meanings, Google Translate is able to correctly translate it into "彼はテーブルの上に本を置いて部屋を出た。". On the other hand, Google Translate misinterprets the word "*bank*" in the sentence "A fisherman is sitting on the **bank**." and thus translates it into "漁師が銀行に座っています。". The cause of this problem can be hypothesized that machine translation models use only one embedding vector to represent a homograph, even though the senses of a single homograph can be completely unrelated. Consequently, if a machine translation model fails to understand the meaning of a homograph from the context, it will tend to choose the dominant meaning. Furthermore, another hypothesis is that the parallel corpus used to train Google Translate system does not provide enough information for the system to understand the context of the latter example. This problem of semantic ambiguity can be addressed if both following conditions are satisfied:

1. Different senses of a homograph are represented by different embedding vectors. To achieve this, the model needs to understand the meanings of homographs from the context.

2. The training set must be exceptionally large so that the model can properly understand the context of unseen sentences. Although it is possible to obtain a large monolingual data set, it becomes difficult when it comes to finding a parallel corpus.

4 Contextualized word embeddings from BERT

4.1 BERT

An extensive pretrained language representation model, BERT supports transfer learning and fine-tuning on a wide range of tasks, such as question answering and language inference(Devlin et al., 2018). BERT is composed of multiple layers of bidirectional Transformer encoders, with 12 layers for $\text{BERT}_{\text{BASE}}$ and 24 layers for $\text{BERT}_{\text{LARGE}}$ model(Devlin et al., 2018). As a result, BERT can learn bidirectional word representations by conditioning on both left and right contexts, which outperforms unidirectional language models, such as ELMo(Peters et al., 2018) and OpenAI-GPT(Radford et al., 2018).

BERT is simultaneously pretrained on two different tasks: *masked language modeling* and *next sentence prediction*. As for the *masked language modeling* task, 15% of all tokens in each sequence are selected at random to be predicted. If a token is chosen, it is replaced with a [MASK] token 80% of the time, with a random token 10% of the time or with the same token 10% of the time. BERT learns to predict the masked tokens instead of regenerating the entire input. As for

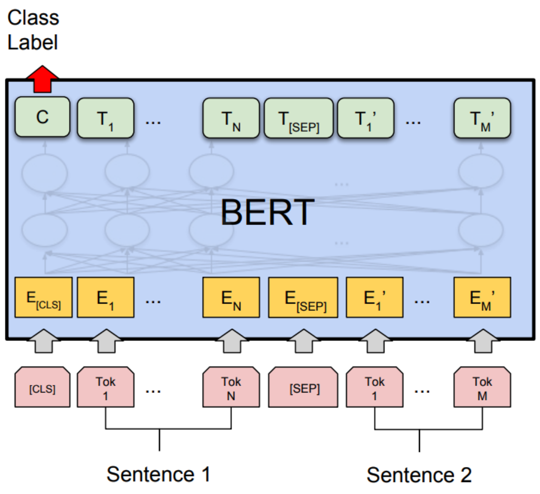

Figure 2: The BERT architecture

the *next sentence prediction* task, BERT is trained with a collection of concatenated sentence pairs and tries to predict whether the two sentences in each pair are contiguous. Consequently, the pre-trained BERT is capable of understanding the relationship between two sentences and can be fine-tuned on downstream tasks, such as *question answering* and *natural language inference*(Devlin et al., 2018).

The corpora used for pretraining BERT are the BooksCorpus (800M words) and English Wikipedia (2,500M words). BERT uses WordPiece embeddings(Wu et al., 2016) with a vocabulary of 30,000 tokens. Each sequence input to BERT starts with a [CLS] token followed by two concatenated sentences, which are separated by a special [SEP] token.

4.2 Contextualized word embeddings

Since the pretrained BERT generates the vector representation for a word by considering all other words in the same sequence, this vector can change dynamically with the context where it is used. In other words, BERT can produce different embeddings for different senses of the same word. To examine this feature of BERT, a pre-trained BERT was used to generate the vector representations of words in different examples:

1. "There is an old man fishing on the **bank**."

2. "It's on the north **bank** of the Thames."

3. "a house on the **banks** of the River Severn"

4. "He jumped in and swam to the opposite **bank**."

5. "Many of these **banks** issue both credit and debit cards."
6. "My salary is paid directly into my **bank**."
7. "A group of ten international **banks** is to underwrite and sell the bonds."

The words *bank* and *banks* in the first four examples mean "*the edge of a river*," while the ones in the next three examples mean "*a financial institution*." After word embeddings of dimension 768 are extracted from the pretrained BERT, t-SNE algorithm(Maaten and Hinton, 2008) is used to extract the two most significant features of each word and the reduced word embeddings are visualized as shown in Figure 3. It can be clearly seen that the points representing the words "*bank*" and "*banks*" are clustered in two separate groups based on their meaning. Furthermore, another interesting point is that the words "*bank*" and "*banks*," which mean "*the edge of a river*" are located near related words such as "*river*," "*fishing*," and "*swam*," while the ones meaning "*a financial institution*" are near to some monetary terms such as "*credit*" and "*mortgage*."

5 Context-aware machine translation architectures

In the original Transformer architecture, each word in the predefined vocabulary list is represented by only one embedding vector. These word embeddings are trained as the model's parameters, therefore depend greatly on the limited training set. Apparently, the original Transformer does not satisfy the conditions mentioned in section 3. Consequently, it is unable to correctly translate semantically ambiguous sentences, as demonstrated in section 6.3

The $\mathbf{BERT_{BASE}}$ model was integrated into a Transformer translation model to address lexical ambiguity in neural machine translation. Specifically, two architectures are proposed: $\mathbf{Transformer_{BERT-WE}}$ using the pretrained BERT as input word embedding layer and $\mathbf{Transformer_{BERT-Encoder}}$ replacing the encoder stack of a Transformer with the pretrained BERT model. The outputs of the last ten layers of the BERT were extracted and averaged. All the parameters of the pretrained BERT were kept unchanged during the training phase of both models.

In this work, we implement and compare the performance of three models: a baseline Transformer, a $\mathbf{Transformer_{BERT-WE}}$, and a $\mathbf{Transformer_{BERT-Encoder}}$, which share the same hyperparameters for comparison purposes. We denote the number of layers in both the encoder and decoder stacks as N, the dimension of embedding layers and all sublayers' output as d_{model}, the dimension of the inner layer in every fully connected feed-forward layer as d_{ff}, and the number of attention heads as h. Due to the lack of memory capacity, N is set to 3, as opposed to $N = 6$ in the original Transformer paper. In addition, to match the hidden size $h = 768$ of the pretrained BERT model, d_{model} and d_{ff} are set to 768 and 3072 respectively. Dropout is applied to the output of each sublayer and the sum of the embeddings and positional encodings in both the encoder and decoder with a rate of 0.1.

6 Experiments

6.1 Data

The models are trained with Japanese–English Subtitle Corpus (JESC)(Pryzant et al., 2018), consisting of over 3.2 million sentence pairs. This data set includes casual language, colloquialisms, expository writing, and narrative discourse. The train/val/testsplits are of size 3,237,374 / 2000 / 2001.

Rakuten MA (morphological analyzer) is used to tokenize Japanese sentences and tokens that appear at least 10 times are shortlisted. Likewise, as for the baseline Transformer model, English sentences are tokenized by using nltk library and tokens are shortlisted in the same manner. By contrast, for BERT Transformer models, the BERT's internal word tokenizer and vocabulary are used. All out-of-vocabulary words are replaced with a special [UNK] token. For the pretrained BERT to effectively generate vector representations, special tokens [CLS] and [SEP] are added to the beginning and the end of the input English sentences, respectively.

Over 6,000 sentences that contain homographs are extracted from the IWSLT 2017 English–Japanese data set to evaluate the performance of the models on ambiguous sentences. The Bilingual Evaluation Understudy (BLEU) score is used as the evaluation metric to assess the models' performance.

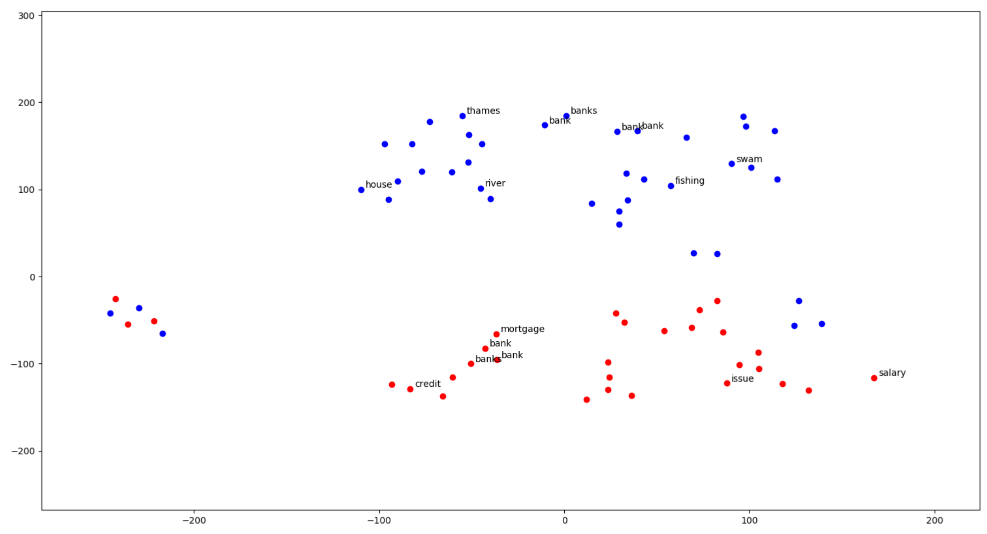

Figure 3: Visualizing word representations generated by BERT using t-SNE.

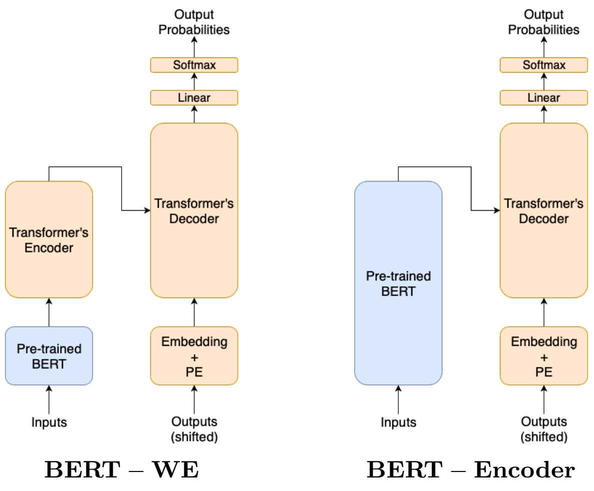

Figure 4: Proposed context-aware machine translation architectures integrating a pretrained BERT and a Transformer model.

6.2 Training details

The three models were trained with the same training settings. The batch size was set to 32 and AdamOptimizer(Kingma and Ba, 2014) with decayed learning rate used. The initial learning rate is set to 0.0001. The models are evaluated on a validation set after each training epoch. When validation loss increases for the first time, the learning rate starts decreasing with the rate of 0.33 per epoch. The training process is stopped when validation loss increases again. It takes from 1.5 to 2 days to finish training a model on a GTX 1080Ti.

6.3 Results

The models were evaluated based on two criteria: general translation and homograph translation. The JESC test set was used to evaluate the models' capability of translating English sentences in general, while the IWSLT 2017 data set was used to evaluate the models' ability to correctly translate semantically ambiguous sentences.

The results of BLEU score evaluation are shown in Table 1. It can be clearly seen that the $\textbf{Transformer}_{\textbf{BERT}-\textbf{WE}}$ model outperforms the other two models in both evaluations, achieving a BLEU score of 20.31 on JESC test set and 8.67 on IWSLT 2017 data set. $\textbf{Transformer}_{\textbf{BERT}-\textbf{Encoder}}$ model's performance is slightly worse than the vanilla Transformer in terms of general translation; however, it outperforms the vanilla Transformer when it comes to translating homographs.

As shown in Table 2, Google Translate and the vanilla Transformer wrongly translate the word "*bank*" in the two given English sentences. By contrast, the two models $\textbf{Transformer}_{\textbf{BERT}-\textbf{WE}}$ and $\textbf{Transformer}_{\textbf{BERT}-\textbf{Encoder}}$ can correctly translate the word "*bank*" into "土手" or "岸", which means "the edge of a river" in Japanese.

	JESC (general)	IWSLT 2017 (ambiguity)
Transformer	18	7.23
Transformer$_{\text{BERT-WE}}$	20.31	8.67
Transformer$_{\text{BERT-Encoder}}$	17.78	8.16

Table 1: BLEU score evaluation of three translation models on two different test sets

Source sentences	1. A fisherman is sitting on the **bank**.
	2. He is swimming to the opposite **bank**.
Google Translate	1. 漁師が銀行に座っています。
	2. 彼は反対側の銀行に泳いでいます。
Transformer	1. 漁師が銀行に座っています
	2. 彼は正反対の銀行に向かって泳いでいます
Transformer$_{\text{BERT-WE}}$	1. 漁師が土手に座っている
	2. 反対側の岸に泳いでいる
Transformer$_{\text{BERT-Encoder}}$	1. 漁師が土手に座っています
	2. 彼は反対側の岸に向かって泳いでいる

Table 2: Sample translations produced by different machine translation systems. We highlight the homographs in source sentences in bold, the corresponding wrongly translated words in red and the corresponding correctly translated words in blue and green.

7 Related Work

Another approach to solving lexical ambiguity was proposed by Liu et al. (2018)(Liu et al., 2017). The authors proved that the performance of translation models degrades as the number of senses for each word increases. They concatenated embedding vectors from a word sense disambiguation (WSD) system to a translation model's word embeddings and applied gating functions to generate contextualized word embeddings. The contextualized word embeddings are fed into the encoder of an RNN-based seq2seq model to generate translations. Their model was trained on three different language pairs: English–German, English—French, and English–Chinese.

Using pretrained word embeddings was empirically proved to increase the BLEU score for the machine translation task. Qi et al. (2018)(Qi et al., 2018) compared the performance of different translation systems that used either random initialization or pretraining on both source and target languages. The word embeddings used in their experiments were trained by using the Common Bag of Words (CBOW) algorithm(Mikolov et al., 2013). According to their results, using pretrained word embeddings, especially on the source language side, considerably boosts the performance of machine translation systems(Qi et al., 2018).

8 Conclusion

In this work, we demonstrate that lexical ambiguity is an inherent problem that contemporary machine translation systems cannot completely address, hypothesize two causes of the problem, and prove that this issue can be addressed by using contextualized word embeddings that dynamically change based on the context of given words. In addition, the BERT language model is demonstrated to be effective at generating contextualized word representations and two machine translation architectures that integrate pretrained BERT and Transformer translation models are proposed. The two architectures are shown to be able to translate semantically ambiguous sentences effectively. Furthermore, the **Transformer$_{\text{BERT-WE}}$** model outperforms the vanilla Transformer model, proving that our approach can not only resolve the problem of lexical ambiguity, but also increases the translation quality in general.

References

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer normalization. *arXiv preprint arXiv:1607.06450*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N Dauphin. 2017. Convolutional sequence to sequence learning. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1243–1252. JMLR. org.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Frederick Liu, Han Lu, and Graham Neubig. 2017. Handling homographs in neural machine translation. *arXiv preprint arXiv:1708.06510*.

Minh-Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*.

Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.

R. Pryzant, Y. Chung, D. Jurafsky, and D. Britz. 2018. JESC: Japanese-English Subtitle Corpus. *Language Resources and Evaluation Conference (LREC)*.

Ye Qi, Devendra Singh Sachan, Matthieu Felix, Sarguna Janani Padmanabhan, and Graham Neubig. 2018. When and why are pre-trained word embeddings useful for neural machine translation? *arXiv preprint arXiv:1804.06323*.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding with unsupervised learning. Technical report, Technical report, OpenAI.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI Blog*, 1(8).

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

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