Neural Machine Translation

Seminar Report-NLP
Arul, Ehsan
Rheinische Friedrich-Wilhelms-Universität Bonn
July 19, 2017

***Abstract*—Abstract**

I. Introduction

II. Deep Learning - An Introduction

In the recent years, Deep learning has been making a big impact in various fields of computer science. This impact is profound in the perceptual learning task in the fields like computer vision, natural language processing, etc, [13]. Though the idea of training a multi layered neural network to perform function approximation is known to the research community for at least a couple of decades[16], due to the nature of training problem requiring large computational resources, the multi layered neural network remained less practical. With the advent of general purpose graphical processing units(GPGPUs) and the availability of training data, the neural networks are now a practical solution for many perceptual tasks. One of the early success of deep learning was in the field of image classification. In the annual ImageNet classification challenge [5], AlexNet [12] showed a remarkable improvement in the state-of-the-art accuracy. Within a few years, more sophisticated architectures like Google Inception Network [18], Deep Residual Network [6], etc., has improved the accuracy to be comparable with a human in that task. The generality of the neural network made it easily possible to be used for a wide variety of tasks. With more people working on deep learning and the ideas arising in solving on the problems being easily transferable, the deep learning is the state-of-the-art for many learning tasks. In the following section, we will briefly summarize the basics of deep learning.

The basic building block of a Multilayer network, or in more technical term Multilayer perceptron(MLP) is a perceptron. The preceptron 2 closely resembles a neuron in a human brain 1. A preceptron takes a set of input values, computes a weighted sum of the inputs, and outputs a scalar value of a after applying an activation function (mostly non-linear). The weights for computing the weighted sum and the threshold in the activation function are initially set to random values and are learned from the training data. Mathematically, a perceptron with the weight matrix A and the threshold T for an step activation function, for the input vector x, outputs the following,

$$f(x) = \begin{cases} 1, & \text{if } Ax > c \\ 0, & \text{otherwise} \end{cases}$$

A layer has many number of neurons and many layers are stacked one on top of the other to form a MLP. An MLP with very many layers is called as Deep Neural Network(DNN). In the following section we will discuss the properties of DNNs and how they are useful in the context of supervised machine learning.

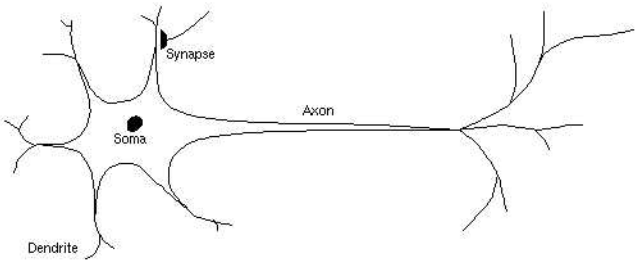


Fig. 1: A biological neuron.

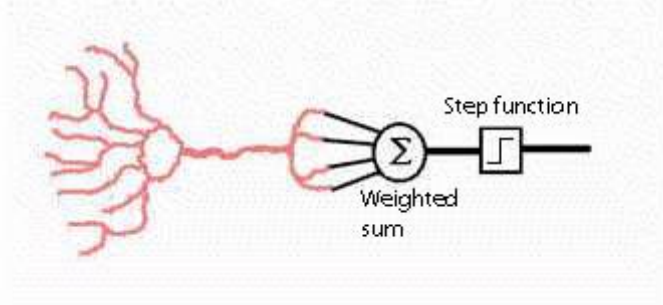


Fig. 2: An artificial neuron (perceptron)

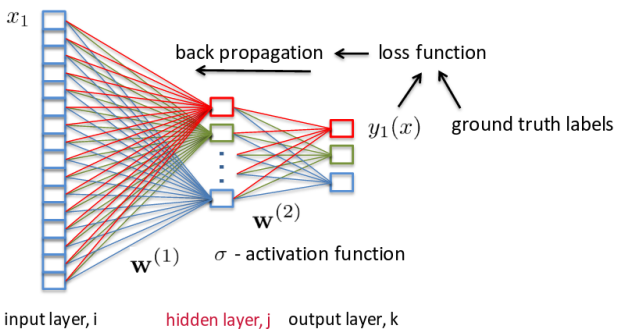


Fig. 3: Layered representation in an MLP

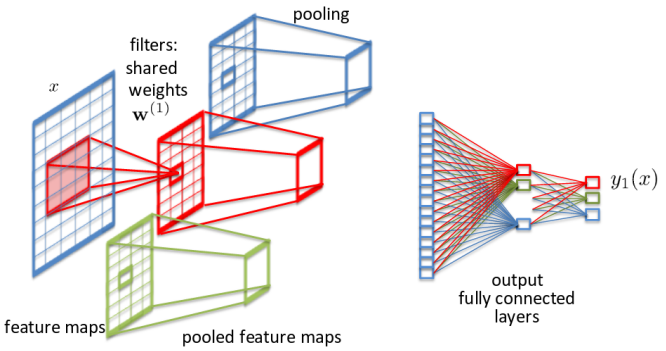


Fig. 4: CNN

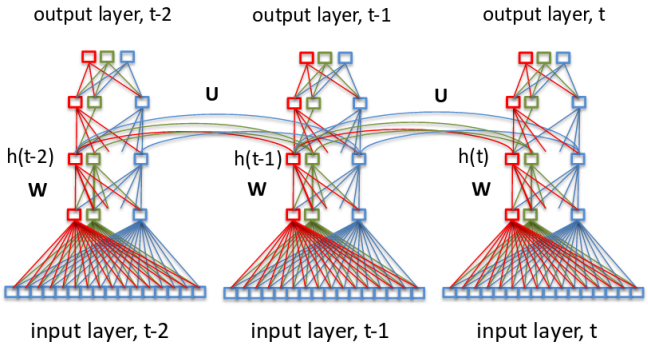


Fig. 5: Recurrent connections in a RNN

A. *Supervised learning*

Supervised learning is the problem of predicting an new output $y\prime$ for a new input $x\prime$, and set of training data that provides a set of input and output $\{x \mapsto y\}$. Most of the supervised learning problems can be formulated as either a classification or regression problem. Classification is the problem of predicting the label for a given $x\prime$ among a set of given labels $\{Y\}$ while regression of predicting a continuous valued output for a given input. The usual way of doing supervised learning is to extract features from the given data and train a model to perform classification or regression.

$$x \xrightarrow[\text{Feature extraction}]{\mathbb{D}(x)} x\prime_{intermediate} \xrightarrow[\text{classifer/regressor}]{\mathbb{F}(x\prime|\theta)} y$$

The power of the DNNs lie in the fact that the models learns the features directly from the given training data and avoids hand engineered features.

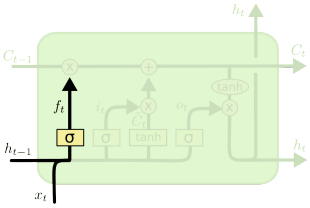
$$x \xrightarrow[\text{hierachical}]{\mathbb{M}(x|\Theta)} y$$

The DNNs with neurons in one layers being connected to all the neurons in the next layer are called Feed Forward Networks. The feed forward networks are harder to train since they have lots of parameters. To make a the DNNs learn useful representation for performing supervised learning, we need special types of connections between the neurons. Two of the most commonly used DNN architectures are Convolutional Neural Networks(CNN) and Recurrent Neural Networks(RNN). In the following section we will discuss these architectures in detail.

1) CNN: todo

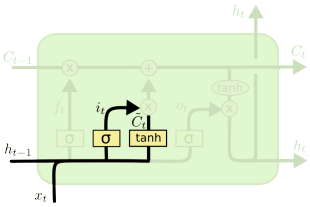
The Convolutional NN are shown in 4

2) RNN: The neurons in the RNNs have recurrent self connections. i.e.The output of the neurons are connected back to the inputs. Thus the output at time step t_1 is computed with taking output at time step t_0 as input. The recurrent connections are shown Fig.5 This enables the RNNs to have an internal memory. RNNs are trained with the a modified version of back propagation called **Back Propagation Through Time(BPTT)** [19].



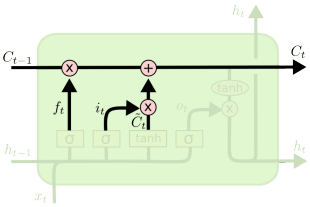
$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

Fig. 6: Forget gate in LSTM



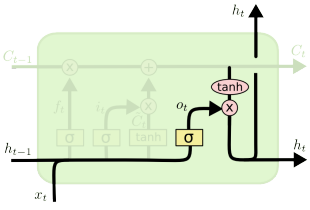
$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

Fig. 7: Add gate computing parts to be to modified in LSTM



$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

Fig. 8: Add gate computing data to be added to cell state in LSTM



$$o_t = \sigma\left(W_o \; [h_{t-1}, x_t] \; + \; b_o\right)$$
$$h_t = o_t * \tanh\left(C_t\right)$$

Fig. 9: Add gate computing output in LSTM

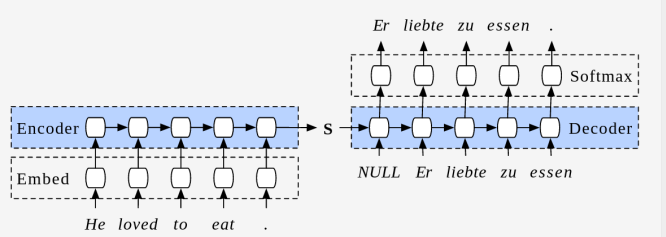


Fig. 10: Simple Encoder-Decoder architecture for machine translation

RNNs in general are harder to train and this is particularity evident when the BPTT is done over a larger number of time steps. This is because the gradients simply converges to zero after a few time steps. This problem known as **vanishing gradients problem** is a well studied phenomenon [3].

To alleviate the vanishing gradients problem, a special architecture called **Long Short Term Memory(LSTM)** [8] is used. An LSTM cell is shown in .

LSTM cells has separate internal memory vector and has three gates–forget gate, add gate, and output gate. To understand the Mathematical intuition behind these gates, let us consider a LSTM cell in a hidden state h and internal cell memory vector c_{t-1}. At time t, the cell takes input x_t, previous output h_{t-1} and update the cell memory to c_t, produces the output h_t. The forget computes a vector size c between 0 and 1. This vector determines who information should be preserved and what should be forgotten based on the current input x_t as shown in 6. This vector is later multiplied with the cell memory. If the vector is all zeros, multiplying makes all cell state to forget everything while all ones preserve everything. Then the add gate computes which parts in the cell states to be updates as shown in 7 and the new data to be updated as shown in 8. Finally the output gate generates the output h_t and does not modify the cell state as shown in 9.

B. Evaluation score

Translation is a hard task to define an evaluation score to measure how well a model is performing due to inherent complexity of the task. The most widely used evaluation metric is called Bilingual Evaluation Understudy(BLEU)[15]. Depends on modified n-gram precision (or co-occurrence). Needs lots of target sentences for better results. Despite being the most widely used metric, many researchers have expressed concern about the effectiveness of the metric [22], [4], [1]. Consider the following translation candidates

- Candidate 1: It is a guide to action which ensures that the military always obey the commands the party.
- Candidate 2: It is to insure the troops forever hearing the activity guidebook that party direct.

to be evaluated against the set of three reference translations.

- Reference 1: It is a guide to action that ensures that the military will forever heed Party commands.
- Reference 2: It is the guiding principle which guarantees the military forces always being under the command of the Party.
- Reference 3: It is the practical guide for the army always to heed directions of the party.

The BLUEs core idea is to use count the number of N-gram matches. The match could be position-independent. Reference could be matched multiple times. These steps are linguistically-motivated.

Candidate 1: It is a guide to action which ensures that the military always obey the commands of the party.
Reference 1: It is a guide to action that ensures that the military will forever heed Party commands.
Reference 2: It is the guiding principle which guarantees the military forces always being under the command of the Party.
Reference 3: It is the practical guide for the army always to heed directions of the party.
N-gram Precision : 17

Candidate 2: It is to insure the troops forever hearing the activity guidebook that party direct.
Reference 1: It is a guide to action that ensures that the military will forever heed Party commands.
Reference 2: It is the guiding principle which guarantees the military forces always being under the command of the Party.
Reference 3: It is the practical guide for the army always to heed directions of the party.
N-gram Precision : 8

Thus candidate 1 is better. **Issues with N-gram precision**

Candidate: the the the the the the the.
Reference 1: The cat is on the mat.
Reference 2: There is a cat on the mat.

N-gram Precision : 7 and BLEU : 1

This result is very misleading. Thus the following modified BLEU score is often used. N-grams with different Ns are used

Algorithm	Example
Count the max number of times a word occurs in any single reference	Ref 1: The cat is on the mat. Ref 2: There is a cat on the mat. âtheâ has max count 2
Clip the total count of each candidate word	Unigram count = 7 Clipped unigram count = 2 Total no. of counts = 7
Modified N-gram Precision equal to Clipped count/ Total no. of candidate word	Modified-ngram precision: Clipped count = 2 Total no. of counts =7 Modified-ngram precision = 2/7

but 4 is most common metric.

C. Phrase based Machine Translation(PBMT)

Traditionally there exists two types of machine translation systems:

- **The Rule-based Approach** The source language text is analyzed using parsers and/or morphological tools and transformed into intermediary representation. This representation is used to generate target sentence. The rules are written by human experts. As a large number of rules is required to capture the phenomena of

natural language, this is a time consuming process. As the set of rules grows over time, it gets more and more complicated to extend it and ensure consistency.

- **The Data-driven Approach** In the data-driven approach, bilingual and monolingual corpora are used as main knowledge source. In the statistical approach, MT is treated as a decision problem: given the source language sentence, we have to decide for the target language sentence that is the best translation. Then, Bayes rule and statistical decision theory are used to address this decision problem.

Phrase based Machine Translation following the paradigm of Data-driven Approach Statistical Machine Translation achieved impressive result and were widely scalable models in the beginning of the millennium. [11]. The model works at the level of phrases instead of words and had a lot of individual components but the core idea is learning statistical patterns in training data.

Some of the major components used in the PBMT were

- Sentence alignment: Gale and Church Algorithm based on Dynamic programming [**?**].
- Word alignment: Expectation Maximization.
- Phrases generation: Heuristic based complex algorithms.
- Phrase lookup: Statistical matching.
- Beam search: For generating target sentence. Beam search is a generic algorithm that is used even in the latest NMT systems.

Beam search in general is a heuristic search algorithm that explores a graph by expanding the most promising node in a limited set. In machine translation, beam search is used to generate a set of most likely sentence given a input sentence by retaining only the most promising translations. Number of sentence is kept constant.

D. Neural Machine Translation(NMT)

Though usage of neural networks did not yield promising results in the early years, the recurrent neural networks started to achieve performance comparable to PBMT as in [10] and [7]. Most of the early architectures were simple encoder-decoder architectures. An simple working of encoder-decoder architecture for machine translation is shown in Fig.10. The architecture has an encoder that takes a sentence in source language and encodes it into a vector S of fixed length. The decoder takes the embedding S as input and generates the sentence in the target language. Some of the major drawbacks with the simple architecture is that

1) The encoder has to encode all the information in the source sentence into fixed size embedding S.
2) The decoder never sees the actual input sentence and has to rely completely on S for generating target sentence.
3) Having fixed size embedding vector makes the architecture less flexible. Smaller size means less information where as using larger vector means for smaller sentences we need zero padding and more computation time.

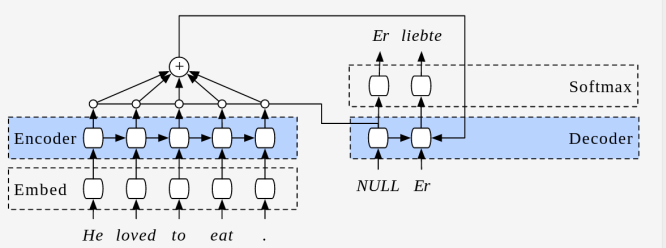


Fig. 11: Context for machine translation

1) Jointly learning to align and translate: The paper [2] address this issue of having a fixed size embedding by introducing the idea of context 11. The main proposal was to use

1) Encoder outputs a hidden representation for each word in the source sentence F_s.
2) One context vector C in the size of the input sentence that has values between 0 and 1.
3) Each element of the embedding F_s is multiplied with one element in C.

The whole embedding is made available to the decoder and C describes which part of the embedding should be focused to generate the current word based on the previous word.

Encoder RNN, at each input step t, generates hidden state, $h_t = f(x_t, h_{t-1})$.

Unlike in the previous models where only the last hidden state is made available to the decoder, this paper provided all the hidden state and also a context vector that has a weight for each of the hidden vector. The context vector helps the decoder to focus on a part of the sentence. $c = q(\{h_1, \cdots, h_{T_x}\})$.

The decoder is trained to predict the next work y_t given the context vector c and all previously predicted words $\{y_1, \cdots, y_{t-1}\}$

$$p(y) = \prod_{t=1}^{T} p(y_t | \{y_1, \cdots, y_{t-1}\}, c)$$

With RNN, each conditional probability is modeled as,

$$p(y_t | \{y_1, \cdots, y_{t-1}\}, c) = g(y_{t-1}, s_t, c)$$

where s_t is the hidden state of the RNN. The context vector for a input sentence i, is computed as a weighted sum of hidden states of the encoder (also known as **annotations**)

$$c_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j$$

$$\alpha_{ij} = \frac{exp(e_{ij})}{\sum_{k=1}^{T_x} exp(e_{ik})}$$

where,

$$e_{ij} = a(s_{i-1}, h_j)$$

is the alignment model that scores how well the inputs around the j and the output at the position i match.

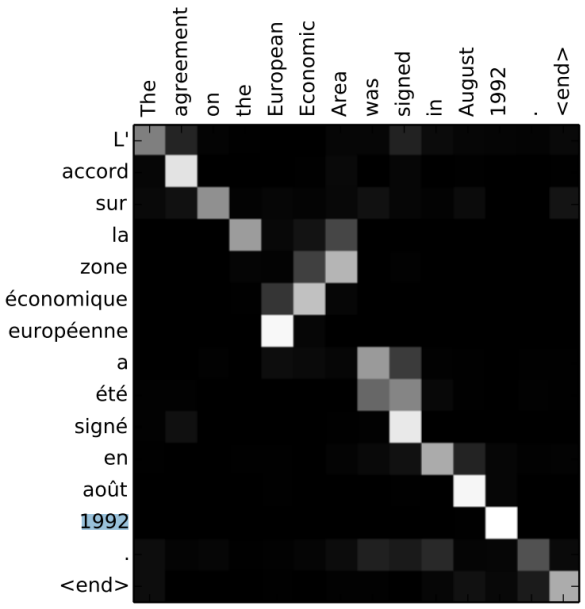


Fig. 12: Visualization of the context in action

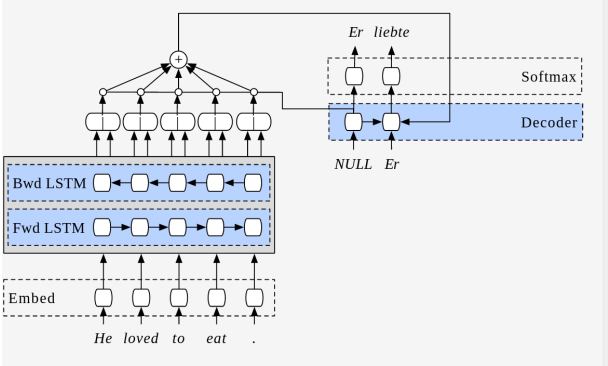


Fig. 13: Bi-directional Encoder

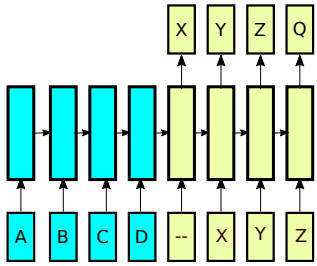


Fig. 14: Seq2Seq model

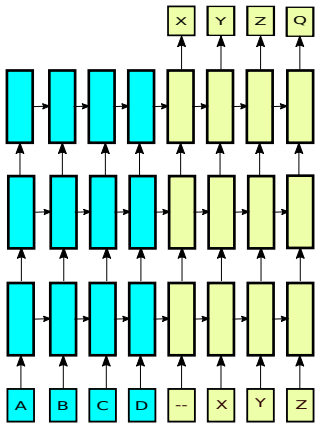


Fig. 15: A more powerfuls Seq2Seq model

A feedforward neural network is used as the alignment model and is **jointly trained** with all the NMT system as a whole.

The functionality of the context vector is visualized in the Fig. 12. For example, the French language has the words "European Economic Area"exists in the reverse order as "zone économique européenne". The context learns to focus on the correct order for the decoder.

This paper also made use of the bi-directional LSTM in the layers of the Encoder. Just like an LSTM that has an internal memory to comprehend the past states, the LSTM also comprehends the words that comes next in the sentence. It has two internal state—one for past states and one for the future state. A bi-directional LSTM is shown in 13.

The whole model is trained with standard maximum-likelihood error minimization $\mathbb{O}_{ML}(\Theta) = \sum_{i=1}^{N} logP_{\Theta}(Y^{*(i)}|X^{(i)})$ with stochastic gradient descent(SGD) on a mini-batch of 80 sentences. For the first time, the BLEU score was comparable to phrase based machine translation system. But the fact that there are no individual pieces in the model was a great benefit and the research community started to consider neural systems as an viable alternative phrase based machine translation system.

2) Sequence to Sequence models.: With in an year, more deep and powerful models were computationally possible. Taking the availability of the computational power to advantage a new framework for learning sequence to sequence mapping were proposed. A sequence to sequence model formulated as

$$p(y_1, \cdots, y_T|x_1, \cdots, x_T) = \prod_{t=1}^{T} p(y_t|v, y_1, \cdots, y_{t-1})$$

where v is the internal memory of the RNNs. With more powerful RNNs, this model can be used to learn a variety of tasks like speech recognition, handwritten digits recognition, machine translation, etc as shown in the paper [17]

A simple sequence to sequence model is shown in Fig.15. Just by making the model more deeper, without any special treatment for machine translation, the paper achieved state-of-the art results. The model as trained on WMT English to French dataset with 12M sentences consisting of 348M

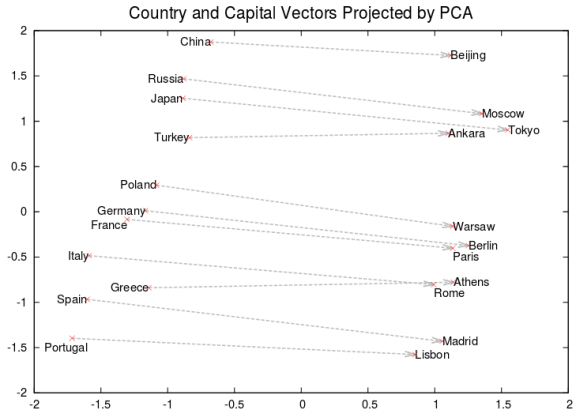


Fig. 16: Two-dimensional PCA projection of the 1000-dimensional word embedding.

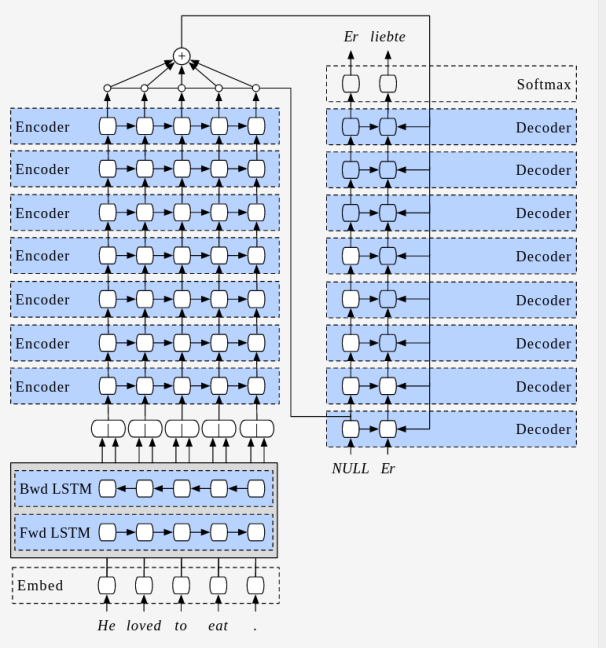


Fig. 17: The core architecture of GNMT

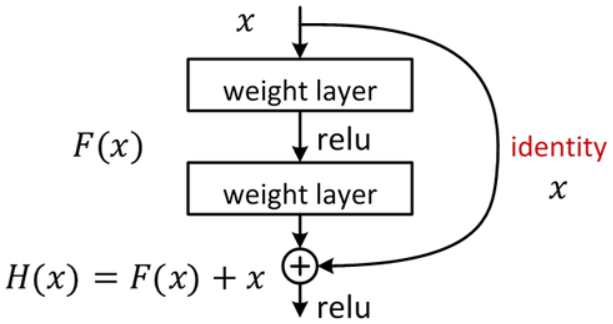


Fig. 18: Residual connections

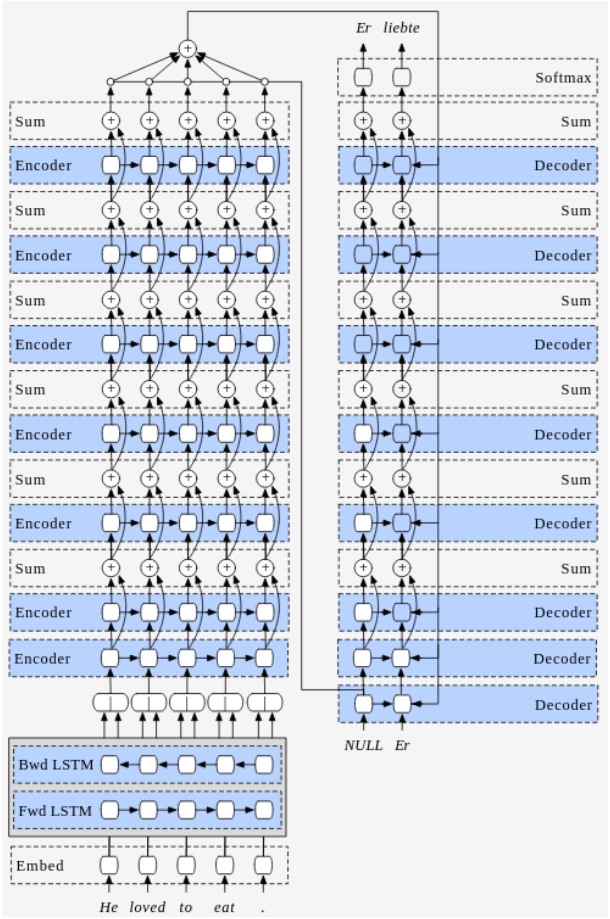


Fig. 19: GNMT with residual connections

French words and 304M English words using 160,000 of the most frequent words for the source language and 80,000 of the most frequent words for the target language. Every out-of-vocabulary word was replaced with a special **UNK** token. The network has 4 LSTM layers with 1000 LSTM cells in each layer. The paper used 1000 dimensional word embedding to represent the words as vector following the word2vec paper [14]. word2vec formulated the problem of learning word embedding as an energy maximization problem using a simple neural network with just one hidden layer. The energy maximized is called Negative sampling. The resulting vector embedding is empirically shown to have arithmetic properties. i.e. France - Paris + Germany is roughly equal to Berlin as shown in Fig.16. The paper an impressive BLEU score of 33.3 even without doing anything specific for machine translation.

3) Google Neural Machine System: Most likely Google runs the worlds biggest [20] machine translation service supporting 103 languages(at the time of writing this article) serving about a billion query every day. Google translate team published a detailed paper describing their neural machine translation system [21]. In the following section we will discuss the key ideas described in the paper. This paper is very unique in the sense that it is much more than an academic paper. The paper explains most of the components of the system including a lot of engineering details and also has a lot of components that are inspired from other recent breakthroughs in machine translation and deep learning research

in general. The basic architecture is heavily inspired by the jointly learning to align and translate II-D1 and II-D2. The core architecture is shown in Fig.17. It combines the idea of context with the a deeper sequence to sequence encoder-decoder model with bi-directional LSTM in the initial layers of the encoder. Both the encoder and decoder contains eight layer. It also made use of the Residual connections introduced in [6]. Residual connection are helpful in alleviating the vanishing gradients problem in an interesting manner as shown in 18. In a DNN each layer can be interpreted as transforming the input x to a new manifold by learning $F(x)$ but in a residual network only learn the change to be applied to the input $(x + F(x))$. There is always an identity skip connection between layers helping in the constant gradient flow which otherwise might vanish. The residual learning enables training very deep networks as shown in Fig.19. The author also tried to use a enhanced cost function. The usual cost function maximized by the training process is the maximum likelihood

$$\mathbb{O}_{ML}(\Theta) = \sum_{i=1}^{N} logP_{\Theta}(Y^{*(i)}|X^{(i)})$$

but this does not directly correspond to the BLEU score. To improve the BLEU the authors proposed the following cost function.

$$\mathbb{O}_{ML}(\Theta) = \sum_{i=1}^{N} \sum_{Y=\mathbb{Y}}^{N} logP_{\Theta}(Y^{*(i)}|X^{(i)})r(Y, Y^{*(i)})$$

where $r(\Delta)$ is per-sentence score computed as an expectation over all Y up to certain-length.

In terms of the training process, just one encoder and one decoder is used for all language pairs. The authors reported several nice properties of the joint language training. One highlight is that this enables zero-shot learning. The system is able to translate between the language pairs it never saw in the training data. Also the languages for which huge training data exists benefited from the joint training. The zero shot learning property is discussed in detail the paper [9]. To enable a single decoder to generate target sentence for all the languages, the input text is suffixed with additional tokes like $< \_\_EN\_\_ >, < \_\_FR\_\_ >, < \_\_DE\_\_ >, < \_\_ES\_\_ >$ indicating the target language to be generated.

REFERENCES

[1] R Ananthakrishnan, Pushpak Bhattacharyya, M Sasikumar, and Ritesh M Shah. Some issues in automatic evaluation of english-hindi mt: more blues for bleu. *ICON*, 2007.

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

[3] Yoshua Bengio, Patrice Simard, and Paolo Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166, 1994.

[4] Chris Callison-Burch, Miles Osborne, and Philipp Koehn. Re-evaluation the role of bleu in machine translation research. In *EACL*, volume 6, pages 249–256, 2006.

[5] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 248–255. IEEE, 2009.

[6] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

[7] Karl Moritz Hermann and Phil Blunsom. Multilingual distributed representations without word alignment. *arXiv preprint arXiv:1312.6173*, 2013.

[8] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[9] Melvin Johnson, Mike Schuster, Quoc V Le, Maxim Krikun, Yonghui Wu, Zhifeng Chen, Nikhil Thorat, Fernanda Viégas, Martin Wattenberg, Greg Corrado, et al. Google's multilingual neural machine translation system: enabling zero-shot translation. *arXiv preprint arXiv:1611.04558*, 2016.

[10] Nal Kalchbrenner and Phil Blunsom. Recurrent continuous translation models. In *EMNLP*, volume 3, page 413, 2013.

[11] Philipp Koehn, Franz Josef Och, and Daniel Marcu. Statistical phrase-based translation. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 48–54. Association for Computational Linguistics, 2003.

[12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[13] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

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

[15] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics, 2002.

[16] Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural networks*, 61:85–117, 2015.

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

[18] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2818–2826, 2016.

[19] Paul J Werbos. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560, 1990.

[20] Wikipedia. Google translate, 1999.

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

[22] Ying Zhang, Stephan Vogel, and Alex Waibel. Interpreting bleu/nist scores: How much improvement do we need to have a better system? In *LREC*, 2004.