Neural Machine Translation Techniques Used By Google

Sophia K. Mitchellette
Division of Science and Mathematics
University of Minnesota, Morris
Morris, Minnesota, USA 56267
mitc0417@morris.umn.edu

ABSTRACT

Google's Neural Machine Translation System employs multiple techniques in conjunction to improve translation accuracy and efficiency. In this paper we summarize two of the techniques used by Google in their Neural Machine Translation System: the attention network model and the residually connected network model.

The attention network model extends the encoder-decoder network model by adding an attention mechanism. The attention mechanism allows for proper encapsulation of information for longer sentences; the amount of information given to the decoder network is based on the length of the input sentence. The attention network model outperforms the encoder-decoder model and provides more accurate translations for longer sentences.

Plain deep neural networks have connections only between adjacent network layers; adding more layers to this kind of network eventually results in a peak accuracy, after which adding more layers degrades network accuracy. The residually connected network model extends the plain network model by adding residual connections between network layers. With residual connections, networks continue to improve with an increased number of layers, instead of peaking and then degrading in accuracy.

1. INTRODUCTION

The goal of a translation is finding the most likely conversion of the *source language*[1] sentence to the *target language*[2] sentence. In translating an English sentence to French: English is the source language and French is the target language.

Machine Translation (MT) is the attempt to automate the process of finding the most likely translation of a source language sentence into a target language. Statistical Machine Translation (SMT) was a type of MT more commonly used in the past, but Neural Machine Translation (NMT) has become more prominent over time. Google previously used a phrase-based SMT system, and they have since switched to Google's Neural Machine Translation (GNMT) system. Neural MT networks have been making progress towards achieving and exceeding the success rates of SMT systems. Google applied multiple improvement techniques that brought their GNMT system to a higher success rate than Google's previous phrase-based SMT system. [14]

We describe two of those techniques in this paper. We explain how each technique works and how Google made use of them in their GNMT system. The first technique we describe is the attention network model. The second technique is the residually connected network model. For both of these techniques, we describe the results of the initial evaluations done by the techniques' inventors.

[1]This is the language you are translating *from.*
[2]This is the language you are translating *to.*

UMM CSci Senior Seminar Conference, December 2017 Morris, MN.

2. NEURAL NETWORKS

Neural networks are composed of *nodes* [1]. Nodes *map* inputs to outputs [1]. A mapping is synonymous with a function. One can also say that nodes are functions which take in inputs and yield outputs. In MT, words and sentences are often represented as vectors [3]. In the case of vector representation, the vector representation of the source language text would be the input to the neural network [3]. The vector representation of the target language translation would be the output of the neural network [3]. Sentences may be segmented on different bases; each sentence segment is represented by a vector [14].

Word segmentation is a type of segmentation which breaks down sentences based on words; each word would be represented by one word vector [14]. Word vectors may represent certain features of words, such as their meaning [9]. In this case, words similar in meaning have a closer position in vector space [9]. The significance of word vector positions are dependent on the type of vector representation chosen. *Character segmentation* is another type of segmentation which represents each character in the sentence with one vector [14]. Another type of segmentation is used by GNMT called *wordpiece segmentation*; the sentence segmentation is based on maximizing the probability of achieving the best translation [14]. In Google's example sentence "Jet makers feud over seat width with big orders at stake." The word "feud" is represented by two vectors, one vector for "fe" and another for "ud" and the word "width" is represented by one vector because that is the segmentation that maximizes the probability of a correct translation.

Neural networks learn through training to produce a desired output based on an input. For instance, in MT, neural networks are given text in a source language, and the neural network ideally returns the most likely translation of that text in the target language. [3]

Neural networks are organized into *layers*. There are three

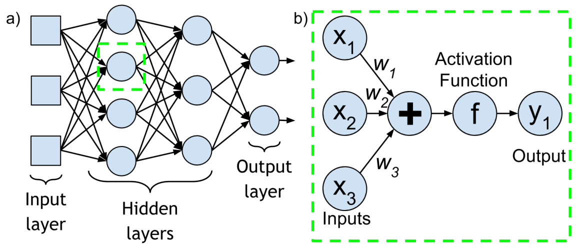

Figure 1: a) A diagram of a neural network. The arrows between nodes represent network connections. [1] b) An example of a single node in the overall network. Here, y_1 could be a number representing data.[2]

sections of layers: the *input layer*, the *hidden layers*, and the *output layer*. The input layer is the first layer of a network, and is comprised of the network's input values. The output layer is the last layer of the neural network, and the hidden layers are the layers in between the input layer and the output layer. The output layer and the hidden layers are comprised of nodes. There can be many hidden layers or no hidden layers. [1]

Neural networks have *connections* between nodes [1]. If some or all of node u's outputs are inputs to node k, and node u is from a layer before node k, then we refer to the nodes as having a *forward connection*. If some or all of node u's outputs are inputs to node k, and node u is from a layer after node k, then we refer to the nodes as having a *backward connection*.

If a network has one or more hidden layers, then the input to the first hidden layer is the input layer. The nodes of the i^{th} hidden layer are connected to the nodes of the $(i+1)^{th}$ hidden layer. The nodes of the last hidden layer are connected to the nodes of the output layer. The output of the network is the output of the nodes of the output layer. If the network has no hidden layers, then the input layer is the input to the nodes of the output layer. Figure 1a is a diagram of a neural network with two hidden layers. The first hidden layer has four nodes, and the second hidden layer has three.

A network is considered to be a *feed-forward neural network* if the network has only forward connections between layers, and not backward connections or connections between nodes in the same layer [11]. Another kind of multiple layer neural network where nodes on the same layer are connected –called a *Recurrent Neural Network*–is covered in Section 3.

2.1 Node Structure

A simple network node has three components. In the first component of a node, the node's inputs are multiplied by their respective weights. An input with a higher weight has more impact on the node's output. An input with a weight of zero would have no impact on the node's output. In the second component of a node, the weighted inputs are added together. In the third component of a node, the sum of the weighted inputs is passed through an *activation function*. The result of the activation function is the output of the node. [1]

Figure 1b is a diagram of one of the nodes in Figure 1a.

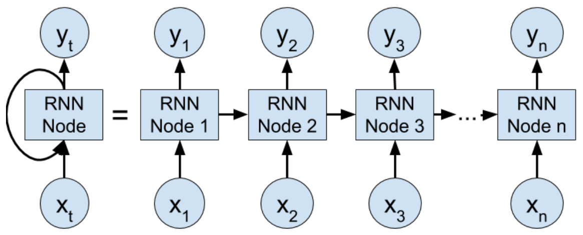

Figure 2: Example of single layer RNN [4]. On the left is the RNN with looping, on the right is the RNN looping unrolled. [4]

This node takes in the inputs x_1, x_2, and x_3. The summation step is represented by the circle containing a "+". The node's activation function is f. The result of the activation function is y_1. The second hidden layer's nodes will receive y_1 as input through the node's forward connections.

Activation functions are usually non-linear, as this introduces non-linearity into the neural network. Non-linearity in a neural network allows the network to learn more complicated data patterns. If all of the activation functions in the network are linear, then the network is unable to learn more complex data. Common activation functions include the hyperbolic tangent function ($tanh$) and sigmoid (σ). Activation functions such as $tanh$ and σ scale the output values of nodes to an interval[3], which prevents their absolute values from growing too large. [13]

The respective weights of each input are calculated through training the network. Neural networks are trained by giving them data sets [1]. For MT, a data set could contain vector representations of paired source and target language sentences [3]. During training, the neural network tries different weight values in an attempt to minimize its *error*; the error is the difference between the actual output and the desired output [1]. That is to say, the neural network is trying to bring its actual outputs closer to the desired outputs. A networks *training error* is the networks error when run over the data it was tested on. The networks *testing error* is the networks error when run on new and unfamiliar data.

3. LSTM RNNs

In an Recurrent Neural Network (RNN), layers are repeated in a loop, and each *time step* becomes its own layer [4]. A time step in a neural network is an instance of evaluation. For MT there is one time step for each sentence segment [3]. When using word segmentation: For the sentence "They run fast.", the words "They", "run", and "fast" would each be a time step [3]. An example of an RNN node is given in Figure 2.

Long Short-Term Memory networks (LSTMs) are a type of RNN. LSTMs are composed of LSTM units. LSTMs solve the *vanishing gradient problem* which occurs with plain RNNs. [11]

The vanishing gradient problem is caused by the activation function condensing the numerical range to an interval [12]. For instance, consider $tanh$. Regardless of how large the absolute value of x is, $tanh(x)$ will lie in the interval of (-1, 1). In this manner, values are condensed over the

[3]$tanh$ and σ restrict outputs to (-1, 1) and (0, 1), respectively.

course of multiple layers (and therefore many runs through an activation function). As a result, in a plain RNN, with each additional step, the impact of earlier steps lessens and the information from earlier steps has less impact [11].

LSTMs avoid the vanishing gradient problem plain RNNs have through the internal structure of their LSTM units. The internal structure of LSTM units are composed of multiple nodes [11]. Three of these nodes are referred to as *gates* [11]. The three gates control how much information from previous layers is maintained, and how much is lost as it leaves the unit [12]. The input gate determines amount of input to be kept and maintained [12]. The output gate determines amount of information in the unit that the unit will output [12]. The forget gate determines what information the unit will forget [12].

If an LSTM layer is *bidirectional* it has both forward and backward connections. Being bidirectional allows the network the advantage of looking at the context of segments before and after the current time step; each time step corresponds to the translation of one source sentence segment. As a result, bidirectional LSTMs have the advantage of looking at the context of the sentence surrounding the current segment being translated. [14]

4. ATTENTION NETWORK MODEL

Attention networks extend a simpler approach known as the encoder-decoder model. The typical encoder-decoder model has only two networks: an encoder network and a decoder network. The encoder network takes in the *input segments*: the vector representations of the source sentence segments. Then the encoder network outputs a fixed length vector which contains all of the information for translating the sentence. The role of the decoder network is to provide a translation based on the vector. The decoder network outputs the *output segments*: the vector representations of the target sentence segments. [3]

Encoder-decoder networks suffered from issues in translating long sentences. Because the vector given to the decoder network is fixed-length, a three-word sentence has as much information given to the decoder network as a fifty-word sentence. As a result, the translation of long sentences has a higher rate of error; not enough information about longer sentences is encapsulated in the vector given to the decoder network. [3]

The attention network model extends the encoder-decoder model by adding an *attention mechanism*. Attention networks are composed of three neural networks: an encoder network, a decoder network, and an attention network. In Bahdanau et al. and the GNMT systems, the attention network is feed-forward neural network. The networks are connected by the attention mechanism, which is the interaction between the three networks. The encoder and decoder networks implemented by Google and Bahdanau et al. were both composed of types of RNNs. In the attention network model, the encoder network produces a series of vectors containing information about the source sentence. The decoder network uses those vectors to produce a translation, and the attention mechanism determines the impact those vectors have on the translation. In the attention network model, the amount of information for a sentence is dependent of the length of that sentence. The decoder network gets more information for longer sentences and less information for shorter sentences. A visual of the attention network

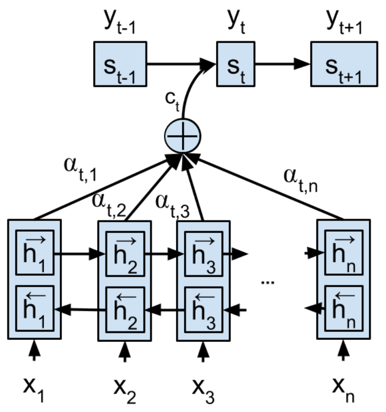

Figure 3: The overall structure Bahdanau et al.'s attention network. This shows the translation of the t^{th} word of a sentence with n words.

as implemented by Bahdanau et al. is given in Figure 3; the figure's annotations will be explained in Sections 4.1 and 4.2. [3]

4.1 Encoder Network

The encoder network takes in the source sentence and produces a sequence of hidden state vectors. There is one hidden state vector for each input segment. So if word segmentation is used, then there is one hidden state vector for each word of the input sentence. If wordpiece segmentation is used then each wordpiece would have its own hidden state vector. [3]

The hidden state vectors contain information about the entire sentence, but with focus on their respective segment, and the parts of the input sentence surrounding their segment. That is to say that the hidden state vector h_j contains information about the entire input sentence, but focuses on the j^{th} input segment and the parts of the input sentence surrounding the j^{th} input segment. [3]

Google's encoder network is comprised of eight LSTM layers, with a bidirectional first layer. Bahdanau et al. has a single bidirectional layer comprised of *gated hidden units*. Like LSTM Units, gated hidden units are composed of multiple nodes and solve the problem of lost information. Gated hidden units differ from LSTM units in that they have two gates, a reset gate and an update gate, in comparison to the three gates that LSTM units have. [3]

The bidirectional layer produces two outputs at each time step, the forward hidden state vectors from the forward connections, $\overrightarrow{h}_j$, and the backward hidden state vectors from the backward connections, $\overleftarrow{h}_j$ [3]. These are concatenated into a single vector, $[\overrightarrow{h}_j, \overleftarrow{h}_j]$ [3]. In the GNMT system, the concatenations are the inputs to the second decoder network layer, while for Bahdanau et al. they are the outputs of the decoder network.

4.2 Decoder Network

Google has eight LSTM layers for its decoder network, while Bahdanau et al. used one gated hidden unit layer. The last layer of Google's decoder network and the only layer of Bahdanau et al.'s decoder network pass through the *softmax* function. The *softmax* function returns the word vector for the word in the target language which has the highest probability of being the correct translation [14].

To produce each segment of the output at time step i, the decoder network takes three inputs. The first input is c_i, the context vector of the segment being translated. The second input is y_{i-1}, the vector representation of the previously translated segment; it is also the output of the $softmax$ function at the previous time step. The third input is s_{i-1}, a hidden state vector of the previously translated segment produced by the decoder network. [3]

For Google, s_{i-1} is the output of the first layer of the decoder network from the previous time step. For Bahdanau et al. s_{i-1} is the output from their only decoder layer from the previous time step.

There is one context vector per output segment. At time step i, the decoder network would use the context vector c_i to translate the output segment, y_i. [3] The context vector c_i, is calculated by multiplying each of the input's hidden state vectors by their respective weight for that time step. The vector representation of the j^{th} input segment is x_j. The hidden state vector h_j is the hidden state vector for x_j. The weight of h_j at time step i is α_{ij}. The context vector for the i^{th} segment of the output would be:

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

The value of α_{ij} reflects the impact of x_j on the translation of y_i. A higher weight value would mean a greater impact. For example, if using word segmentation: At time step i the greatest weight may be assigned to the hidden state vector of the input word that directly translates into the output word being produced. The second highest weight might correspond to the word that impacts the conjugation of the i^{th} output word. [3]

4.3 Attention Mechanism

The attention mechanism produces the weight for each hidden state vector. At time step i, the weight for the j^{th} input segment is α_{ij}. The attention network is part of the attention mechanism, and it evaluates how well the inputs neighboring the j^{th} segment and the output at time step i match. The variable e_{ij} is the output of the attention network and is directly correlated with α_{ij}. The value of α_{ij} is produced by the following equation:

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

The function $exp(x)$ means e^x and the variable n is the total number of input segments. The attention network takes two inputs to calculate e_{ij}. One input is h_j, the hidden state vector of the j^{th} input segment produced by the encoder network. The other input is s_{i-1}, a hidden state vector of the previously translated segment produced by the decoder network. The variable s_{i-1} is defined in Section 4.2. The variable e_{ij} is defined as:

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

The attention network is represented by a. [3]

4.4 Evaluations of Attention Networks

Model	BLEU Score
RNNencdec-30	13.93
RNNencdec-50	17.82
RNNsearch-30	21.50
RNNsearch-50	26.75

Table 1: The BLEU scores for the evaluated networks on ACL WMT '14. RNNencdec is the encoder-decoder network, RNNsearch is the attention network. The 30 or 50 indicates the maximum sentence length the network was trained on. [3]

Google does not provide evaluations of their network with and without an attention mechanism. Instead, we give a summary of the results of Bahdanau et al. To evaluate their network, Bahdanau et al compared their attention network to an encoder-decoder network. For their testing, they used the ACL WMT '14 dataset, which contained 3003 sentences that hadn't been in their training set[4]. Both the encoder-decoder network and the attention network had the 30,000 most commonly used words for training. Both models were trained twice: first on sentences up to 30 words long, and then on sentences up to 50 words long. [3]

The networks were evaluated with BLEU scoring [3]. In BLEU scoring, a higher score means a closer correspondence to human translations [10]. BLEU scores are computed based on how many words, and groups of words are the same between human translations and the translation of the MT system [10]. BLEU scores range from 0 to 1, although in Table 1 they are converted into percentages. As is seen in Table 1, both versions of the attention network outperformed either version of the encoder-decoder networks.

5. RESIDUALLY CONNECTED NETWORKS

5.1 Difficulties in Training DNN

Deep Neural Networks (DNN) are composed of more than one hidden layer. The *depth* of a network refers to its number of layers. A network with more layers is a deeper neural network than a network with fewer layers. [2]

Plain neural networks have connections only between adjacent network layers. Increasing the depth of a plain neural network eventually results in a peak accuracy, after which increasing the network depth degrades network accuracy. Increasing the plain neural network depth prior to the peak in accuracy has another benefit; it allows the neural network to account for more complex patterns in data. [6]

Google found that in their experience simple stacked LSTM layers work well up to 4 layers, barely with 6 layers, and very poorly beyond 8 layers with large-scale translation tasks [14]. He et al. compared two plain neural networks, a 20-layer network and a 56-layer network. When running the trained network back over the training data, the 56-layer network had a higher error both over the training data and unfamiliar testing data than the 20-layer network [6].

The deeper network having a higher error may seem unexpected given that there is a way to ensure that a deeper network can always perform at least as well as a shallower

[4]The ACL WMT '14 dataset was data provided by the Association for Computational Linguistics for their 2014 translation task.

counterpart: Consider a neural network with w layers. If v more layers were added onto the network (making it a network of $q = w + v$ layers), then the new q-layered network always has a way to be as accurate as the w-layer network; make the v added layers *identity mappings*, and the first w layers the same as the w-layered network. Identity mappings are mappings in which the output has the same value as the input.[5] [6]

In regards to why the deeper network in their testing had a higher error, He et al. stated that *overfitting* was not likely to be the problem, because the degradation of the network's accuracy was shown in the training data. Overfitting is a problem in which the model learns the training data too precisely, and doesn't learn the overall trend of the data [5]. If overfitting was the cause of the problem, then the data would have done well on the training data and performed poorly on the testing data.

The peak in plain neural network accuracy is the result of difficulty in the training stage, where the system attempts to find the appropriate weights for the network [6]. While initial layers in a neural network may have significant improvements that can be made to their outputs, later layers in a neural network may be close to the desired outputs for all or part of their outputs. The closer the output of the $i+1^{th}$ network layer is to the desired output for the entire network, the closer the mapping of the i^{th} network layer needs to be to an identity mapping. Plain DNN have difficulty in reaching identity mappings and mappings close to identity mappings [6].

Consider the following example to illustrate the difference in finding an identity mapping for a plain network compared to a residually connected network: Given a paragraph of text, perfectly duplicating that paragraph of text would be akin to performing an identity mapping on the paragraph.

If a person represents a node in a plain network, then they are taking the original paragraph as their input. The person in the plain network has to retype the paragraph to duplicate it. The retyping would represent the node's mapping. In retyping the paragraph, there are opportunities for error at every character, (a character could be missed, a different character typed, etc.) Any mistyping would lead to the disruption of an identity mapping. Their resulting paragraph would be the node's output.

If a person represents a node in a residually connected network, then they are taking the original paragraph as their input, and they are copy-pasting the original paragraph. There is far less room for error in copy-pasting. If the person made any edits to the paragraph, then those edits would represent the node's mapping. To achieve a resulting paragraph identical to the original all the person has to do is make no edits.

5.2 The Structure of Residually Connected Networks

The residually connected network model extends the plain network model by adding *residual connections* between network layers [14]. Residual connections pass x_t^i, the input of the t^{th} node in layer i, directly to a summation of x_t^i and s_t^{i+g}, where g is the number of layers between residual connections [14]. He et al has two to three layers of network nodes between residual connections, while Google has only one LSTM layer between connections; see Figure 4 for examples of these structures. Residual connections do not add computational complexity, as they do not add additional parameters in passing along a value. With residual connections, networks continue to improve with increased layers, instead of peaking and then degrading in accuracy [14].

[5]Having identity mappings for the added v layers being the optimal solution is unlikely in real cases.

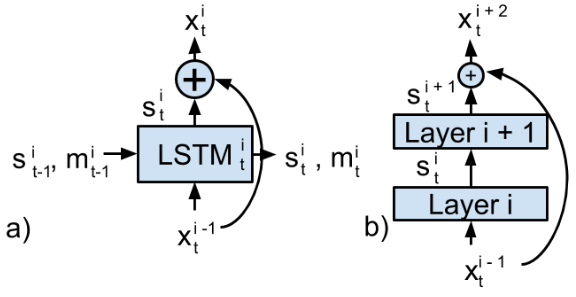

Figure 4: (a) A residually connected LSTM unit as found in GNMT. (b) Part of a residually connected network with two layers between the residual connection as found in He et al. For both, the curved line is a residual connection.

The addition of residual connections in a neural network changes the mappings that the nodes are trying to fit. Plain networks are attempting to directly fit the *desired underlying mapping*. A desired underlying mapping is the actual mapping for a node that renders the optimal output. For instance, if D is the desired mapping for a node in a network, and y is the optimal output for that node, then $D(x_1, x_2, ..., x_m) = y$, where $x_1, x_2, ..., x_m$ are the inputs to the node. Residually connected network nodes are attempting to map to their *residual mapping*, the difference between their desired mapping and their input. For instance, if the desired mapping for a node is D, and the input to the node is x, then the residual mapping for the node is $R(x) = D(x) - x$.

Training for residually connected neural networks benefits from this change in mappings; reaching identity mappings is simpler in a residually connected neural network. In a residually connected network, an identity mapping is where $R(x) = 0$. Given a scenario in which the optimal mapping for a node is an identity mapping, it is likely easier for a node in the residual model to reach an identity mapping of $R(x) = 0$, than it is for a node in the plain model to reach an identity mapping of $D(x) = x$. In order for a node in the residual model to reach an identity mapping of $R(x) = 0$, the node would set its weights to zero. In order for a node in the plain model to reach an identity mapping of $D(x) = x$ it would have to find weight values that would recreate its input, x. [6]

Figure 4 is a look into a residually connected LSTM unit as found in GNMT system. The $LSTM_t^i$ is the LSTM unit at layer i and time step t. The unit's input along the depth of the network is x_t^{i-1}. The units inputs along time are s_{t-1}^i and m_{t-1}^i; s_{t-1}^i is the output of $LSTM_{t-1}^i$ and m_{t-1}^i is the information encoded by $LSTM_{t-1}^i$. The output of $LSTM_t^i$ is s_t^i; this value is the result of the unit's residual mapping. The input to $LSTM_t^{i+1}$ is x_t^i, where $x_t^{i-1} = s_t^i + x_t^{i-1}$ and x_t^{i-1} is provided by the residual connection.

Layers	Non-residual	Residual
18-layer	27.94	27.88
34-layer	28.54	25.03

Table 2: The testing error rates for the networks when looking at the networks actual output against its target output. The result was considered to be in error if the image label produced by the network was not the same as the target image label. [6]

5.3 Evaluations of Residually Connected Networks

The comparisons for GNMT system with and without residual connections are not available. Instead we summarize the evaluations made by He et al. on the effectiveness of residual connections. He et al. created networks for image recognition as opposed to machine translation. The networks evaluated by He et al. were *Convolutional Neural Networks* (CNN), which are commonly used with images. The nodes of CNN take matrix representations of image subsections as their inputs [8]. He et al.'s experiment compared four different CNNs: two residually connected CNN and two plain CNNs. One residually connected CNN is 18-layer, the other is 34-layer. One non-residually connected CNN is 18-layer, the other is 34-layer.

Both residually connected networks outperform either plain network. For plain networks, increasing the network depth from 18-layers to 34-layers increased the networks training error, and as seen in Table 2, increased its testing error as a result. He et al. stated that they, "argue that this optimization difficulty is unlikely to be caused by vanishing gradients" as they used *Batch Normalization* in their neural network. Batch Normalization is a technique that can help to guard against values in the network becoming too large or too small by bringing data into a specified range [7].

In contrast to the results of the plain networks, increasing depth from 18 to 34 layers for the residually connected networks had the opposite effect. For the residually connected networks, increasing the depth led to decreased error, both on training and on testing data, and subsequently decreased testing error. [6]

6. CONCLUSIONS

Attention networks and residually connected networks are two techniques used by Google to improve their GNMT system. Neural networks face issues in maintaining information and encapsulating sufficient information.

Attention networks help to encapsulate sufficient information on sentences by extending the encoder-decoder network model to include an attention mechanism. This enables them to handle long sentences. Residually connected networks help train deep neural networks, which is important because deep neural networks increase a systems accuracy and allow for the network to account for more complex data patterns. Residually connected networks help to maintain information and prevent degradation with increased network layers.

Acknowledgments

Many thanks to my advisor, Elena Machkasova, my professor for senior seminar, K.K. Lamberty, and my alumnus reviewer, Mitchell Finzel, for your guidance and feedback.

7. REFERENCES

[1] Unsupervised Feature Learning and Deep Learning Tutorial. *Computer Science Department, Stanford University.* http://ufldl.stanford.edu/tutorial/supervised/MultiLayerNeuralNetworks/.

[2] Introduction to Deep Neural Networks. *Deeplearning4j*, 2017. https://deeplearning4j.org/neuralnet-overview.

[3] D. Bahdanau, K. Cho, and Y. Bengio. Neural Machine Translation by Jointly Learning to Align and Translate. *CoRR*, abs/1409.0473, 2014.

[4] D. Britz. Recurrent Neural Networks Tutorial, Part 1 - Introduction to RNNs. *WildML*, 2015. http://www.wildml.com/2015/09/recurrent-neural-networks-tutorial-part-1-introduction-to-rnns/.

[5] A. Guillot. Machine Learning Explained: Overfitting. *Enhance Data Science*, 2017. http://enhancedatascience.com/2017/06/29/machine-learning-explained-overfitting/.

[6] K. He, X. Zhang, S. Ren, and J. Sun. Deep Residual Learning for Image Recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[7] S. Ioffe and C. Szegedy. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In *ICML*, 2015.

[8] M. Nielsen. Neural Networks and Deep Learning. *Deeplearning4j*, 2017. http://neuralnetworksanddeeplearning.com/chap6.html.

[9] C. Olah. Deep learning, nlp, and representations. http://colah.github.io/posts/2014-07-NLP-RNNs-Representations/.

[10] K. Papineni, S. E. Roucos, T. Ward, and W.-J. Zhu. BLEU: a Method for Automatic Evaluation of Machine Translation. In *ACL*, 2002.

[11] M. Sundermeyer, H. Ney, and y. v. p. Ralf Schlüter, journal=IEEE/ACM Transactions on Audio, Speech, and Language Processing. From Feedforward to Recurrent LSTM Neural Networks for Language Modeling.

[12] H. Suresh. Vanishing Gradients LSTMs. 2016. http://harinisuresh.com/2016/10/09/lstms/.

[13] A. S. Walia. Activation functions and it's types-Which is better? *Towards Data Science*, May 2017. https://medium.com/towards-data-science/activation-functions-and-its-types-which-is-better-a9a5310cc8f.

[14] Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey, J. Klingner, A. Shah, M. Johnson, X. Liu, Ł. Kaiser, S. Gouws, Y. Kato, T. Kudo, H. Kazawa, K. Stevens, G. Kurian, N. Patil, W. Wang, C. Young, J. Smith, J. Riesa, A. Rudnick, O. Vinyals, G. Corrado, M. Hughes, and J. Dean. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*, 2016.