Neural Machine Translation with LSTM's

Jasjeet Dhaliwal
jdhaliwal@cs.umass.edu

Abstract

Machine translation is a challenging problem mainly due to the inherent structural ambiguities of natural language and the problem of word-sense disambiguation. We tackle the problem of machine translation from English to French by building a neural machine translation system. Inspired by the success of encoder-decoder style neural machine translation systems, we develop an Long Short-Term Memory network based on the same encoder-decoder architecture. We also deploy a state of the art phrase-based statistical machine translation system called Moses, and compare results of both systems on the WMT '14 test dataset using the BLEU (bilingual evaluation understudy) scoring metric. Moses outperforms our machine translation system in all experiments.

1 Introduction

Machine translation is an important domain of research with applications in enterprise systems, educational tools, translation tools, and various other fields. In the past, Statistical Machine Translation (SMT) systems have consistently been the best performing automated translation tools available. However, in recent years, deep recurrent neural network models have been shown to outperform these SMT systems as in [25]. These sequence to sequence models utilize the power of deep neural networks to solve the machine translation task. Recurrent Neural Networks (RNN's), are naturally best suited for this task as it requires learning dependencies between words in sentences as well as learning dependencies between sentence pairs of the source language and the target language. These dependencies are learned well by deep recurrent neural networks because of their ability to maintain a hidden state representation of the words in a sentence, that encodes the meaning of a given sentence. An extension of deep recurrent neural networks are the recurrent neural networks that first encode the meaning of a source language sentence using an encoder module, and then feed this encoded representation to a decoder module, that is responsible for producing the sentence translation in the target language. These encoder-decoder style neural translation systems can be trained end-to-end using optimization methods such as stochastic gradient descent. They do not require the brittle linguistic rules such as those used by traditional RBMT's and phrase-based translation systems such as [18]. However, despite the deployment of successful encoder-decoder style models based on deep recurrent neural networks such as [1, 5, 25, 28], there is lack of a clear consensus on the key properties that allow a such models to successfully learn the complex dependencies of a language, as well as the translation dependencies between the source and target language. As an example, some state of the art encoder-decoder models such as [25] use fixed length vector representations of source sentences whereas other state of the art models such as [5] don't. Along this line of research on using encoder-decoder based neural machine translation systems for solving the machine translation task, we too utilize a variant of RNN's, Long Short-Term Memory (LSTM) networks [11] to build a neural machine translation system. LSTM's have been shown to better learn complex long term dependencies due to their ability to utilize crucial hidden state information over time and ignore less important details. We use LSTM's to build a neural machine translation system that translates sentences from English to French. We train our model end-to-end using stochastic gradient descent on data extracted from the WMT'14 dataset. In addition to training the LSTM network on the machine translation task, one of our primary goals is to identify some key properties that allow such encoder-decoder LSTM networks (and deep recurrent neural networks in general) to learn the complex dependencies of languages. In order to do so we experiment with different network architectures, data preprocessing, language vocabularies, word embeddings, datasets, and training methods.

Our machine translation system is a deep recurrent neural network with Long Short-Term Memory (LSTM) units as introduced by [11]. Based on the encoder-decoder approach, the network is made up of two recurrent neural networks - an encoder and a decoder, each containing LSTM units. The encoder converts varying length English sentences into a fixed continuous space representation. The decoder uses this representation to generate the French translation of the English sentence. Our implementation is based on the Caffe [12] framework and all source code is available here: `https://github.com/jasjeetIM/Seq2Seq`

In order to compare our system to an existing state of the art system, we chose to deploy a phrase-based Statistical Machine Translation (SMT) system, Moses. This choice is primarily based on the dominance of Moses over other open source machine translation tools available. Further, phrase-based SMT systems have dominated other machine translation paradigms such as Rule Based Machine Translation (RBMT), Example Based Machine Translation (EBMT), and Hybrid Machine Translation.

We structure our report as follows. We begin by reviewing related work on machine translation in section 2. We provide the necessary background material on Recurrent Neural Networks, and Long Short-Term Memory networks in section 3. We then describe our dataset, followed by an explanation of our machine translation system in sections 4, and 5 respectively. We analyze our results and outline some key properties that are crucial for learning language dependencies in section 6. Finally, we outline future directions of research in section 7.

2 Related Work

Rule-based machine translation (RBMT) systems were some of the earliest machine translation systems deployed commercially. Systran was one of the most popular machine translation rule based systems that performed translation by peforming the following tasks:

1. parsing the source text
2. analyzing the structure of the source text (via a morphological analyzer for instance)
3. finding a mapping between the source text and the target text
4. removing transfer ambiguity between source and target text
5. flagging grammatical divergences between source and target text such as gender or number agreement
6. creating a sequence of chunks
7. substituting fully-tagged target-language forms into the chunks
8. generating the target text.

However, RBMT systems like Systran required a large amount of time and effort to build. Further, these systems were very brittle due to the hard-coded linguistic rules required by the system.

This is when Statistical Machine Translation (SMT) methods [2] started to gain popularity and have since been used to develop state of the art machine translation systems that have outperformed other paradigms (such as RBMT). To this date, most statistical machine translation systems require a language model and a translation model. The language model helps these translation systems understand the structural dependencies of a language and the translation probabilities help the models map sentences in the source language to sentences in the target language. Most SMT models are trained on large datasets in order to develop statistical weights that favor sentence pairs that potentially have the same underlying meaning. However, these models face issues when translating unseen words and phrases. Therefore, in order to overcome this weakness, SMT models utilize what is commonly referred to as phrase-based translations. [17], developed such a model that estimates how source and target sentences can be generated simultaneously using a joint probability model. This joint probability model can be marginalized to yield separate models for the source to target phrase translation and the target to source phrase translation. [14] extended this work by providing a unifying framework to extract phrase translation pairs and a heuristic based decoding algorithm to find the best possible translation in the search space. This phrase-based translation model translates phrases as atomic units. In their model, the source language sentence is segmented into phrases and each phrase is then translated into the target language. The model maintains a phrase translation table with the probability of every phrase which is used during the decoding process. This many-to-many strategy has been able to translate non-compositional phrases and also use local context in the translation process. Despite having the ability to learn longer phrases with the availability of more data, [14] show that learning phrases longer than 3 words does not significantly impact performance of the translation system. The decoding problem, i.e. finding the best possible target translation is solved by using a heuristic search method such as beam search. This approach greatly simplifies the search problem which can be exponential in the possible translations.

[18] performed comparative experiments between word based models and phrase based models (that are phrase aligned and word aligned) on the Verbmobil Task [27], which is a speech translation task from German to English. They found that phrase based models produced better translation results than the word based models. Given, the prior dominance of phrase-based statistical machine translation systems, we choose to deploy the well supported open source phrase-based statistical machine translation system, Moses [15] in order to compare the results of our method with another state of the art machine translation system.

The potential of Neural Machine Translation (NMT) systems [5, 25] has recently been explored with the goal of improving upon previous machine translation paradigms. The primary strength of NMT systems is their ability to learn complex tasks in an end to end manner. Despite the success of Convolutional Neural Networks (CNN) in image related tasks such as object detection [7, 8, 10, 21] and object segmentation [16, 19, 20], traditional feed forward neural networks such as CNN's have not been able to perform well on natural language processing tasks such as machine translation. The primary reason is their inability to maintain an internal representation of the meaning of a sequence of words. However, this problem has been resolved to a large extent with the use of RNN's, which lend themselves naturally to the sequential word processing tasks such as language modeling, text generation, and machine translation. Perhaps, the first successful use of neural machine translation systems was combining them with a phrase-based translation system in [6] as a separate module to learn phrase representations. These models incorporated a method of estimating translation probabilities that extend to rare and unseen phrases.

Another important step towards the development of successful neural machine translation systems was taken by [23], who extended their previous Phrase-based continuous translation models [24]. They used a neural network to perform the projection of words onto continuous space and used the previous words to predict the probability distribution over the next word. Following in their footsteps, [4] showed that using RNN's in an encoder-decoder style architecture could improve the performance of a statistical machine translation system by using the conditional probabilities of phrase pairs computed by the RNN encoder-decoder as an additional feature in the existing log-linear model. The model was used as a part of a standard phrase-based SMT system by scoring each phrase pair in the phrase table. Another notable attempt at using an RNN based encoder-decoder model to develop a complete translation system was done by [13]. They used the hidden representation of the source sentence to model the conditional probability of the first target word. This word was then fed as input to the model to generate a conditional distribution over the next target word. Hence, their model utilized the same principles as [25] in attempting to capture the long range dependences of a target word on the previous target words, as well as on the source sentence. However, their model was unable to outperform existing

state of the art phrase-based statistical machine translation models.
[25] were perhaps the first to develop a RNN (LSTM) encoder-decoder style neural translation system that outperformed state of the art phrase-based SMT systems on the WMT'14 English to French translation task.

A common problem with the above RNN based encoder-decoder style models is the inability of these models to deal with rare words. [1] proposed a novel mechanism that is geared towards addressing this issue by using an attention mechanism in the decoding process. There have been various other enhancements to the encoder-decoder style neural translation model in order to bring their performance up to par with commercial phrase-based statistical machine translation systems on larger body of texts. We build upon the body of work described above by implementing a deep LSTM encoder-decoder neural network to solve the machine translation task.

3 Background

3.1 Recurrent Neural Networks

Recurrent Neural Networks (RNN) are are special type of neural network designed to process variable length sequential data. Given, an input vector sequence $\mathbf{x} = (x_1, \dots, x_T)$, where each x_i represents the data input at the i^{th} time step, the RNN calculates a hidden state representation and an output for each time step. We denote the hidden state representation at time t as h_t and the output at time t as y_t. Therefore, the RNN calculates $\mathbf{h} = (h_1, \dots, h_T)$ and $\mathbf{y} = (y_1, \dots, y_T)$. The hidden state representation and outputs are calculated using the following equation:

$$h_t = \sigma(W * x_t + V * h_{t-1})$$
$$y_t = U * h_t$$

where σ is the sigmoid function,$*$ denotes the inner product, W, V, U are weight matrices whose values are learned for the specific task at hand by using back propagation through time. Note, that h_0 is required at the first time step and can be set by the user for the particular task. One great advantage of RNN's is the ability to share parameters. Every layer in an RNN shares parameters across time steps. Therefore, the RNN can be seen as having parameters for only one time step, that are shared across various time steps. This greatly reduces the number of parameters that need to be learned, unifies the process of learning complex dependencies, and greatly reduces the time required to learn.

We can view the probability distribution over the outputs $\mathbf{y} = (y_1, \dots, y_T)$ as :

$$P(\mathbf{y}|\mathbf{x}) = \prod_{t=1}^{T} P(y_t|x_1, \dots, x_t)$$

Note that the above distribution makes the conditional independence assumption between the outputs y_i and y_j given $\mathbf{x}$ where $i \neq j$. We can get rid of this conditional independence assumption by feeding the output at time t to the RNN at time $t+1$. This models the following probability distribution over the outputs $(y_1, \dots, y_T)$:

$$P(\mathbf{y}|\mathbf{x}) = \prod_{t=1}^{T} P(y_t|x_1, \dots, x_t, y_1, \dots, y_{t-1})$$

Therefore, RNN's can map a sequence of vectors $\mathbf{x} = (x_1, \dots, x_T)$, to a sequence of outputs $\mathbf{y} = (y_1, \dots, y_T)$. It is easy to note that such RNN's can be very useful for tasks that require mapping a fixed length sequence to another fixed length sequence. However, the task of machine translation requires the ability of a model to map variable length sequences to variable length sequences. That is, the task of machine translation requires the mapping of a sequence of inputs $\mathbf{x} = (x_1, \dots, x_T)$, to a sequence of outputs $\mathbf{y} = (y_1, \dots, y_{T'})$, where T may or may not equal T'.

The encoder-decoder style RNN's address the above issue. They do so by conceptually dividing the network into two modules - an encoder and a decoder. The encoder maps varying length input sequences into a fixed length vector representation. The decoder then uses this fixed length vector representation to generate the variable length target sequence. More specifically, the encoder module of the RNN maps $\mathbf{x} = (x_1, \dots, x_T)$ to $\mathbf{h} = (h_1, \dots, h_T)$ and sets h_T as the fixed length representation of the input sequence. The decoder module then uses h_T to generate $\mathbf{y} =$

$(y_1, \ldots, y_{T'})$. Therefore, the probability distribution over the generated target sequence can be viewed as:

$$P(\mathbf{y}|\mathbf{x}) = \prod_{t=1}^{T'} P(y_t|h_T)$$

where once again the conditional independence assumption can be removed by feeding the output at time t to the RNN at time $t+1$ and yield the distribution:

$$P(\mathbf{y}|\mathbf{x}) = \prod_{t=1}^{T'} P(y_t|h_T, y_1, \ldots, y_t - 1)$$

The above model can be used to develop a encoder-decoder style machine translation system that maps sentences in a source language to sentences in a target language.

3.2 Long Short-Term Memory

The original LSTM architecture was designed by [11] but there have since been many small variations to the cell architecture of LSTM's. The core idea behind LSTM design was to address the long term dependency problem faced by RNN's.

To illustrate this problem, consider the example of a language model predicting the bold word in the sentence: ”The boy saw the apple on the table and ate the **apple**”. The target word ”**apple**”, can be inferred by only looking at the previous 7 words. The language model only needs to look at recent information in order to predict the next word. Now consider the text: ”The boy was allergic to apples and loved candy. He walked over to the kitchen with his favorite candy in his hand. He saw the apple on the table and ate the **candy**”. For the language model to fill in **candy**, it needs to analyze the previous 32 words. There are situations where the model needs more context in order to predict the next word. This context is provided by remembering relevant information from a longer time span in the past. RNN's tend to have problems learning long term dependencies as the length of the sequence to be analyzed increases (i.e. their performance goes down as the length of the sequence containing relevant information increases) as shown in [3]. The LSTM, on the other hand, has the ability to remember long term dependencies while also ”forgetting” some of the history of a sequence when required. This is due to the gated architecture of each LSTM cell which allows it to use some, all, or none of the history from the previous cells and exploit long range dependencies in the sequence. This ability to remember, as well as forget the history in a given sequence makes the LSTM a very powerful tool for many sequence learning problems in NLP. [25] use the LSTM cell architecture proposed by [9]. However, we use the architecture proposed by [30]:

$$\begin{aligned}
i_t &= \sigma(W_{hi} * h_{t-1} + W_{xi} * x_t + b_i) \\
f_t &= \sigma(W_{hf} * h_{t-1} + W_{xf} * x_t + b_f) \\
o_t &= \sigma(W_{ho} * h_{t-1} + W_{xo} * x_t + b_o) \\
g_t &= \tanh(W_{hg} * h_{t-1} + W_{xg} * x_t + b_g) \\
c_t &= (f_t * c_{t-1}) + (i_t * g_t) \\
h_t &= o_t * \tanh(c_t)
\end{aligned}$$

where σ is the logistic sigmoid function. i_t, f_t, o_t, g_t, c_t are the input gate, forget gate, output gate, cell, and cell activations. The weight matrices that learn the parameters of the model are denoted with $W_{\alpha\beta}$, where α, β can be inferred from the above equations.

4 Dataset

4.1 Data Selection

We use the freely available : WMT'14 English - French dataset to train our network. The dataset is made up of pairs of English and French sentences (commonly referred to as bitexts). The complete dataset is made up of the following subsets :

1. ep7 - Europarl version 7 (27.8M)
2. nc9 - news commentary version 9 (5.5M)
3. dev08_11 - old dev dat from 2008 to 2011 (0.3M)
4. crawl - data from common crawl (90M)
5. ccb2 - 10^9 parallel corpus (81M)
6. un2000 - UN corpus (143M)

The above data contains around 850 million French words and is quite noisy. Therefore, it is not advisable to train a neural network directly on this data. The website for the WMT'14 dataset, also provides a filtered version of the above dataset which contains the most "useful" data. This subset of the data contains a total of 12075604 bitext sentences and 348M French words. This dataset is further segmented into the following subsets (where the pc# indicates the percentage of data selected for this subset):

1. ep7_pc45 - Europarl version 7 (27.8M)
2. nc9 - news commentary version 9 (5.5M)
3. dev08_11 - old dev dat from 2008 to 2011 (0.3M)
4. crawl - data from common crawl (90M)
5. ccb2_pc30- 10^9 parallel corpus (81M)
6. un2000_pc34 - UN corpus (143M)

The website provides smaller datasets that were selected after concatenating the entire dataset and using Axelrod's algorithm to score the sentence pairs. The top scoring 2%, 4%, 6%, 8% and 10%, were extracted and have been made available for download. For instance, the top 2% dataset contains 784270 English and French sentence pairs. The average length of a sentence in the top 2% dataset is 19 words in the English text and 18 words in the corresponding French text. We have summarized the data statistics in Table 1. below.

Given, limited access to GPU computing, we select a smaller subset of the dataset to train our models. Listed below are the various subsets that we use to train our models:

1. ep7_pc45 - Europarl version 7 (27.8M)
2. crawl_pc37 - data from common crawl (33M)
3. ccb2_pc15_1- 10^9 parallel corpus (41M)
4. ccb2_pc15_2 - 10^9 parallel corpus (40M)
5. 2% Top Scoring Bitext - (.784M)

For testing, we use the newstest2014 dataset that is provided by the website as well. This dataset contains 3003 English and French sentence pairs. This dataset was also used for testing by [25].

4.2 Singles Dataset

Given that sentences are made of words, we thought it would be beneficial to first train our network on single word conversions. This would allow the network to learn word embeddings that put a word in the source language and its corresponding word in the target language close in the embedding space. Hence, we created a dataset called singles. Each source sentence in this dataset is a single word. We first found a resource:list of about 100,000 most commonly used English words as per Wiktionary.org. Then we utilized, an open source API to convert these source words to their corresponding target translations. The resulting bitext contained 73959 sentences. We then repeated each word in this bitext 7 times (not consecutively) and formed a bitext of 512213 sentences. We trained a subset of our models on this dataset in order to perform ablation experiments.

4.3 Data Pre-processing

In order to feed the data to the network, we needed to perform additional pre-processing steps:

1. We reversed the source sentences. This was a design choice made by [25] as they discovered that the LSTM learned better when source sentences were reversed. A possible explanation for this phenomenon is reversing the words in the source sentence keeps the corresponding words in the target sentence closer to the words in the source sentence, and hence allows to the network to learn the dependencies between corresponding words using back propagation in a more efficient manner.

2. We add a <BOS> symbol the beginning of every source sentence and an <EOS> symbol to the end of every source sentence. This is done prior to reversing the source sentence. The primary inspiration for this comes from the work of [26, 29]. This allows the symbol <BOS> to become a signal to the decoder to start generating the translation.

3. We also add <EOS> to the end of every target sentence during training. In the same vein, during inference, we stop a translation once the <EOS> symbol has been generated.

Avg. Len English	Avg. Len French	Total sentence pairs	Total French Words
19	18	784270	15M

Table 1: 2% Dataset Statistics

5 Method

5.1 Network Architecture

We used deep LSTM's with 4 layers, 1000 cells at each layer. We set the maximum length of the LSTM to be 80 time steps. Hence, the maximum number of time steps that we unroll a network per mini batch is 80. Since the average length of a sentence is less than 19 in the dataset, we assumed that setting and unroll limit of 80 will give sufficient room for the source sentence and the target sentence.

This also means that we pad the source sentences and target sentences with a <PAD> character to meet the 80 time step limit. Similarly, if the total length of the source and target sentence is greater than 80, we truncate whichever is greater than 40 words. Further, specifying the time steps per mini batch is required by Caffe [12]. More specifically, the LSTM layer in Caffe requires a fixed number of time steps per 'stream' to be specified per a mini batch.

Lastly, we learn our own word embeddings in all the models developed.

5.2 Implementation Framework

We implement our network using the Caffe [12] framework. Caffe provides a Matlab interface as well as access to the C++ implementation of each layer in the network. Thus, it provides the developer with complete control over the network and its behavior. This was a primary motivation for us in using the Caffe framework. In addition, we wanted to understand the GPU implementation of network layers which was also afforded to us by Caffe. Finally, the Recurrent layers are newer additions to the Caffe framework and require processing data in a mini batch into time steps and streams (i.e. each sentence in a mini batch is a stream with a fixed number of time steps). This restriction requires that the length of all streams (number of time steps) in a mini batch be specified before setting up the network. Although possibly limiting, we worked around this restriction by setting the maximum time steps of the network to 80 (i.e. length of source sentence plus the length of the target sentence) and were able to train successfully using the Caffe framework.

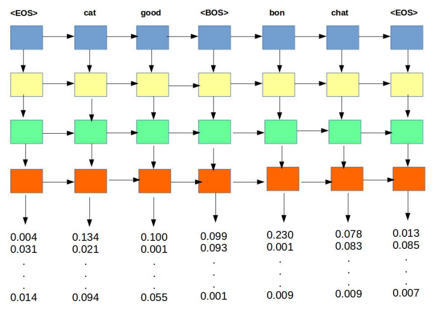
5.3 Vocabularies

We tested two different network architectures based on the vocabularies used. In both cases, we added the following words to the vocabulary:

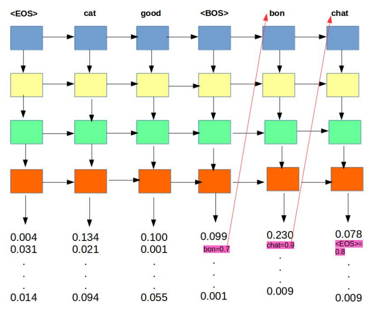
1. <PAD>: Pad source and target sentences
2. <BOS>: Mark the beginning of a source sentence and signal the decoder module to start translating.
3. <EOS>: Mark the end of a source or a target sentence
4. <UNK>: Assigned to words not existing in the vocabulary

Next, we experimented with two different vocabularies:

1. **One vocabulary for source and target language**
   We used a 226,000 word vocabulary that included source and target language words. This vocabulary is provided by the website providing the datasets. The network architecture for this experiment can be seen Figure 1. We trained the network with this architecture on the top 2% dataset and noted that the network assigned high probabilities to the same words every time. Further, these words were common symbols such as ";", "<UNK>". We then trained the network on the ep7_pc45 dataset and got the same results. We think that a possible reason for the poor performance was the large size of the vocabulary (i.e. 226k words). Hence, the network could possibly have a more difficult task in trying to predict the correct word in the target sentence from 226k options, where the options contain source and target words. Due to lack



(a) Train Phase



(b) Test Phase

Figure 1: Network Architecture with one vocabulary. (a) Network architecture during training. The target translations are provided as input to the network. (b) During test time, the highest ranking output from the network is used as the target translation to be provided as input in the next time step. The network stops translating once the <EOS> symbol is generated.

of GPU computational resources, we decided to stop using this network architecture and train a network that utilized two different vocabularies.

2. **Two vocabularies**
   We used two different vocabularies for this network architecture. The source vocabulary consisted of the 150,000 most common words used in the 2% English text and the target vocabulary consisted of the 75,000 most common words used in the 2% French text. In order to train the network, we concatenated the source and the target word embeddings and used the 2000 dimensional embedding as input to the network. Since at any given time, either the source or the target embedding consists of the "<PAD>" embedding, we expected the network to learn this relationship and hence be able to translate from a different source vocabulary to a different target vocabulary. The network architecture with two vocabularies is illustrated in Figure 2.

5.4 Clipping Gradients

One of the general drawbacks of LSTM's is that they tend to suffer from the exploding gradients problem. In order to avoid this issue and to perform ablation experiments, we experimented with enforcing a constraint on the norm of the gradient. That is, letting g be the average gradient of the network for that mini batch, if $||g||_2 > 10$ then set $||g||_2 = 10$. That is we "clipped" the norm of the gradients if their norm exceeded the threshold 10. However, noting that our network was not suffering from the exploding gradient problem. We decided to train two types of models:

1. Models who clipped gradient during training on their 1st dataset
2. Models who never clipped gradients

We provide comparative results for these models in the results section.

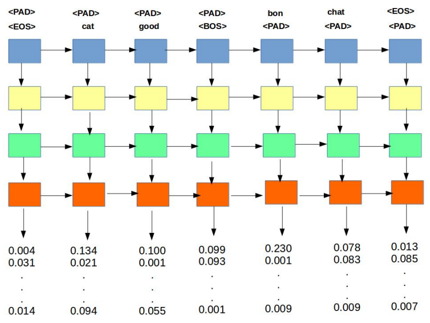
5.5 Decoder

We built a beam search decoder which uses the log probabilities generated by the network as its heuristic. We use a beam width of 4 as the default.

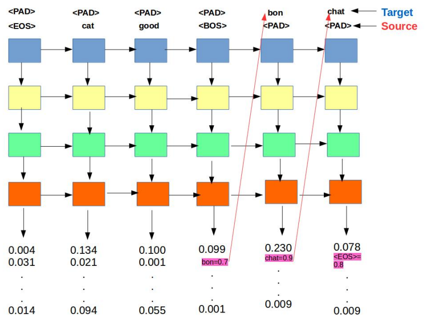
5.6 Training and Loss

We used the following training setup:

1. Initialized all LSTM parameters with the uniform distribution between -0.08 and 0.08.
2. Stochastic Gradient Descent for learning
3. Starting learning rate of 0.7 for every dataset
4. Learning rate decay of 0.1 that was applied at every 10000 iterations.
5. Softmax layer to convert the prediction scores of the network into a probability measure.
6. During training we used the multinomial logistic loss function over the target vocabulary.



(a) Train Phase



(b) Test Phase

Figure 2: Network Architecture with two vocabularies. (a) Source and Target word embeddings are concatenated into one 2000 dimensional vector and provided as input to the network. Once again, the target translations are provided to the network as input during training. (b) This shows how the source and target word embeddings are concatenated as input to the network. The same concatenation logic is used in (a) as well, but it has not be illustrated due to space limitations. During testing, the highest ranking network output is used as input to the network during the next time step.The network stops translating once the <EOS> symbol is generated.

5.7 Models

We enumerate the models trained in Table 2. A check mark indicates that the model was trained on the dataset. A check mark in the CG cell indicates that we clipped gradients on the first dataset that the model was trained on.

We now analyze the training error curve for the different models as follows:

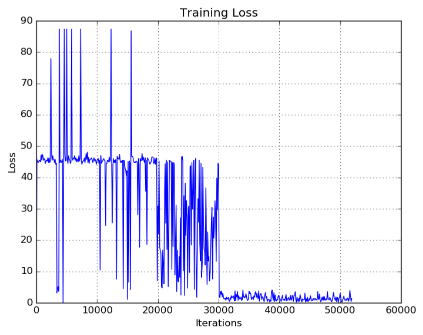
Figure 3 tries to understand whether a model with pre-trained network parameters can learn one to one word mappings. In order to answer this question, we compare the training error curve of two models on the singles dataset. The first is an untrained model with untrained parameters (Net_3) : initialized with uniform distribution between -0.08 and 0.08) and the second is a model with parameters that have already been trained on ccb2_pc15_1 and ep7_pc45 datasets (Net_2). As the figure shows, both curves are very similar and hence we can infer that the network can learn one to one word mapping relationships at any stage of the training pipeline.

Figure 4. attempting to analyze the behavior of the training error curve when we clip gradients versus when we do not clip gradients. The figure plots graphs of two models with untrained parameters on the ccb2_pc15_1 dataset. Once again, we note that both models settle down into a local minimum within approximately same number of iterations. However, upon running the model trained with clipped gradients we note that it learns fewer dependencies than the model trained without clipped gradients. We ascertain this by testing both models on the newstest2014 dataset. The model with clipped gradients outputs a constant symbol ";", where as the model without clipped gradients outputs a more variety of French words. It must be noted that both models output non sensible translations.

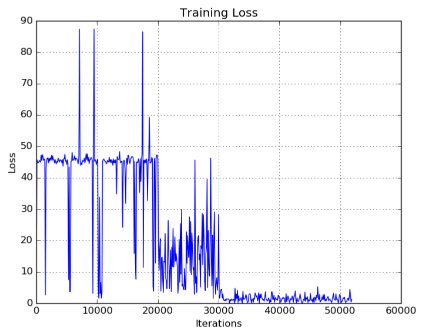
Lastly, we plot the training error curves for all models in Figure 5. These error curves are from the last dataset that the models are trained on. In each model, we note that the training error drops and settles down into a local minima. We discuss this notion of local minima and model performance further in the Results section.

Trained Models						
Name	CG	singles	ccb2_pc15_1	ep7_pc45	ccb2_pc15_2	crawl_pc37
Net_1	✓		✓	✓	✓	✓
Net_2	✓	✓	✓	✓	✓	✓
Net_3		✓	✓	✓		
Net_4			✓	✓	✓	

Table 2: Models Trained

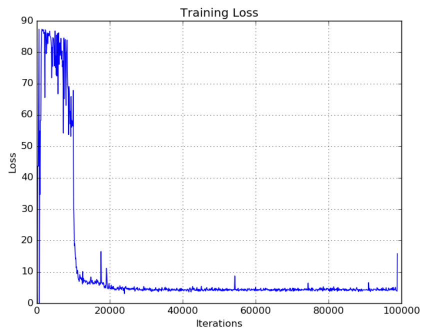


(a) Training on untrained parameters

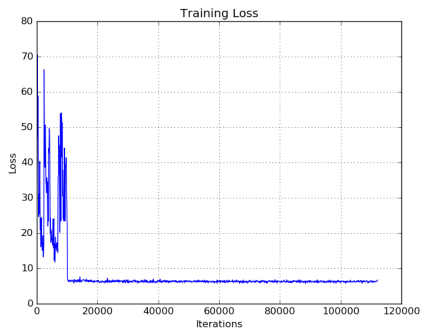


(b) Training on trained parameters

Figure 3: Comparison of the training error curve when training on untrained parameters (a) vs pre-trained parameters (b). We note that the network with untrained parameters initially displays a high training error but gradually settles down into a local minimum. The trained model on the other hand, has a lower training error initially but also finds a similar local minimum.

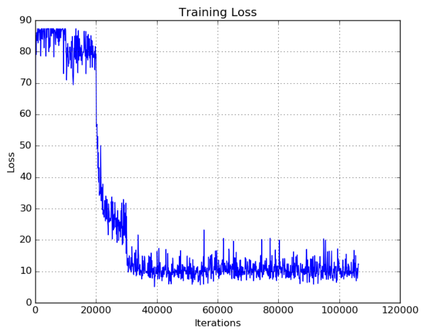


(a) Not clipping gradients

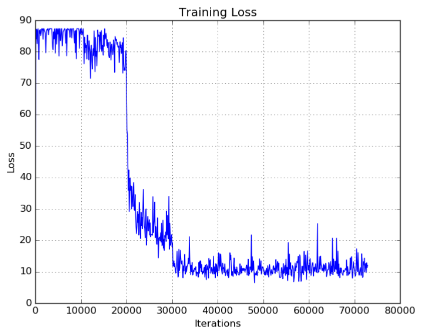


(b) Clipping gradients

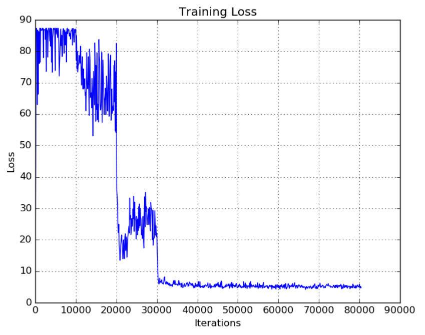
Figure 4: Impact of clipping gradients. Despite the higher initial training error in (a), we see that both (a) and (b) settle in a local minimum in approximately the same number of iterations



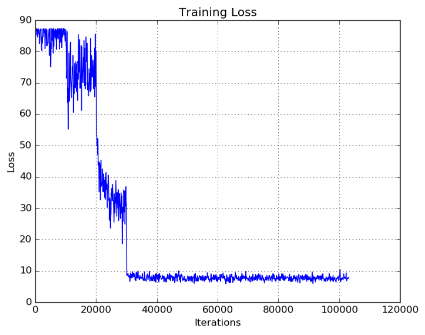
(a) Net_1



(b) Net_2



(c) Net_3



(d) Net_4

Figure 5: Training error curves of all 4 models for the last dataset that they trained on.

6 Results

6.1 Moses

We test the baseline algorithm Moses[15] on the newstest2014 dataset. In order to do so we deploy Moses on our system and use a pre-trained language model for Moses that has been trained on the 2% dataset. This language model was made available by [22] at Moses LM. We use the target text in French as the reference for calculating the BLEU score. We use the multi-bleu.perl script provided in the Moses package [15] in order to calculate the BLEU score. The results are shown in Table 3.

6.2 Our Networks

Our networks do not output syntactically or semantically correct translations. The primary reason for this is that the network parameters settle into a local minimum. We note that while Net_1 and Net_2 memorize the commonly used words in the target language, Net_3 and Net_4 output a large number of ";" and <UNK> symbols. In order for us to get better results, we will need to find the right set of hyperparameters that avoid settling the network into a local minimum. We can address this problem by limiting the network to train for 40000 iterations and using grid search to find a setting of hyper parameters that consistently reduces training error after 20000 iterations. Once the network shows signs of consistently dropping the training error after 20000 iterations, use those hyper parameters to train the network. We can also experiment with gradually clipping gradients in order to fine tune the network after it has learned the fundamental dependencies in the translation. We aim to continue training our network until it outputs sensible translations.

In order to illustrate the network outputs, we display the translation results for the English sentence "Jet makers feud over seat width with big orders at stake" in table 4.

7 Discussion

We implemented a LSTM encoder-decoder style NMT system that converts English sentences into French. In order to better understand the key learning properties of the system, we trained vari-

BLEU Total	BLEU 1-gram	BLEU 2-gram	BLEU 3-gram	BLEU 4-gram
19.68	57.7	28.8	16.1	9.5

Table 3: Moses results on newstest2014

Network outputs	
Gold Std.	Les avionneurs se querellent au sujet de la largeur des siges alors que de grosses commandes sont en jeu
Moses	Jet Chaudronns venimeuse over sige width Live.Reina orders joue
Net_1	la les des en la les des en la les des en la les des en la les des
Net_2	apos les ; le apos le apos des et apos apos apos <UNK> ; la <UNK> ; la <UNK>
Net_3	; les quot)))))))))))))))
Net_4	apos ; ; ; ; ; ; ; ; ; ; ; ; <UNK> <UNK> <UNK> <UNK> <UNK> <UNK>

Table 4: Translation Results by Model

ous models and performed various ablation experiments. We also deployed a state of the art phrase based SMT system and evaluated it on the newstest2014 dataset. Even though our models did not perform well on the translation task, we concluded that this is caused due to the network settling into a local minimum.

We hope to find the correct set of hyper parameters that lead to better translations of the network.

References

[1] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473, 2014. URL `http://arxiv.org/abs/1409.0473`.

[2] Peter F Brown, John Cocke, Stephen A Della Pietra, Vincent J Della Pietra, Fredrick Jelinek, John D Lafferty, Robert L Mercer, and Paul S Roossin. A statistical approach to machine translation. *Computational linguistics*, 16(2):79–85, 1990.

[3] KyungHyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder-decoder approaches. *CoRR*, abs/1409.1259, 2014. URL `http://arxiv.org/abs/1409.1259`.

[4] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

[5] Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *CoRR*, abs/1406.1078, 2014. URL `http://arxiv.org/abs/1406.1078`.

[6] Jacob Devlin, Rabih Zbib, Zhongqiang Huang, Thomas Lamar, Richard M Schwartz, and John Makhoul. Fast and robust neural network joint models for statistical machine translation. In *ACL (1)*, pages 1370–1380. Citeseer, 2014.

[7] Pedro F Felzenszwalb, Ross B Girshick, David McAllester, and Deva Ramanan. Object detection with discriminatively trained part-based models. *IEEE transactions on pattern analysis and machine intelligence*, 32(9):1627–1645, 2010.

[8] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014.

[9] Alex Graves. Generating sequences with recurrent neural networks. *CoRR*, abs/1308.0850, 2013. URL `http://arxiv.org/abs/1308.0850`.

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. In *European Conference on Computer Vision*, pages 346–361. Springer, 2014.

[11] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, November 1997. ISSN 0899-7667. doi: 10.1162/neco.1997.9.8.1735. URL `http://dx.doi.org/10.1162/neco.1997.9.8.1735`.

[12] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*, 2014.

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

[14] 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.

[15] Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondřej Bojar, Alexandra Constantin, and Evan Herbst. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions*, ACL '07, pages 177–180, Stroudsburg, PA, USA, 2007. Association for Computational Linguistics. URL `http://dl.acm.org/citation.cfm?id=1557769.1557821`.

[16] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3431–3440, 2015.

[17] Daniel Marcu and William Wong. A phrase-based, joint probability model for statistical machine translation. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*, pages 133–139. Association for Computational Linguistics, 2002.

[18] Franz Josef Och, Christoph Tillmann, Hermann Ney, et al. Improved alignment models for statistical machine translation. In *Proc. of the Joint SIGDAT Conf. on Empirical Methods in Natural Language Processing and Very Large Corpora*, pages 20–28, 1999.

[19] Pedro O Pinheiro, Ronan Collobert, and Piotr Dollar. Learning to segment object candidates. In *Advances in Neural Information Processing Systems*, pages 1990–1998, 2015.

[20] Pedro O Pinheiro, Tsung-Yi Lin, Ronan Collobert, and Piotr Dollár. Learning to refine object segments. In *European Conference on Computer Vision*, pages 75–91. Springer, 2016.

[21] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.

[22] Holger Schwenk. Continuous space language models. *Computer Speech and Language*, 21:492–518, 2007.

[23] Holger Schwenk. Continuous space translation models for phrase-based statistical machine translation. In *COLING (Posters)*, pages 1071–1080, 2012.

[24] Holger Schwenk, Marta R Costa-Jussa, and José AR Fonollosa. Continuous space language models for the iwslt 2006 task. In *International Workshop on Spoken Language Translation (IWSLT) 2006*, 2006.

[25] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to sequence learning with neural networks. *CoRR*, abs/1409.3215, 2014. URL `http://arxiv.org/abs/1409.3215`.

[26] Subhashini Venugopalan, Marcus Rohrbach, Jeff Donahue, Raymond Mooney, Trevor Darrell, and Kate Saenko. Sequence to sequence – video to text. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2015.

[27] Wolfgang Wahlster. *Verbmobil: foundations of speech-to-speech translation*. Springer Science & Business Media, 2013.

[28] 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.

[29] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *International Conference on Machine Learning*, pages 2048–2057, 2015.

[30] Wojciech Zaremba and Ilya Sutskever. Learning to execute. *arXiv preprint arXiv:1410.4615*, 2014.