Adversarial Attack on Neural Machine Translation System

Dissertation Submitted In Partial Fulfillment Of The Requirements For The Degree Of

Master of Technology
in
Computer Science

by

Abijith K P
[Roll No: CS1730]

Under the Guidance of

Dr. Utpal Garain
Professor
Computer Vision and Pattern Recognition Unit(CVPR)

Indian Statistical Institute
Kolkata-700108, India

CERTIFICATE

This is to certify that the dissertation entitled **"Adversarial Attack on Neural Machine Translation System"** submitted by **Abijith K P** to Indian Statistical Institute, Kolkata, in partial fulfilment for the award of the degree of **Master of Technology in Computer Science** is a *bona fide* record of work carried out by him under my supervision and guidance. The dissertation has fulfilled all the requirements as per the regulations of this institute and, in my opinion, has reached the standard needed for submission.

Dr. Utpal Garain
Professor,
Computer Vision and Pattern Recognition Unit,
Indian Statistical Institute,
Kolkata-700108, India.

Acknowledgment

It was a great privilege and learning experience, to work with Dr. Utpal Garain. He had been a constant source of support, starting from my first year. I want to sincerely thank all the research scholars in NLP Lab. All the discussions I had with Akshay Chaturvedi had influenced me to push my own limits. Thanks to all the friends who had been there with me for the past two years.

Contents

Abstract

Nowadays Deep Neural Network based solutions are deployed to solve numerous tasks. Thus, it has become absolutely important to study the robustness of these systems. Machine Translation is one of the popular applications of Deep Neural Networks. This thesis studies the robustness of Neural Machine Translation systems by generating adversarial examples with the objective to fool the model. Whenever there is a change in the source, i.e. when a word in the input sentence is replaced by an unrelated word, the translation system is supposed to reflect the changes while doing translation. These unwanted invariance learned by the model is undesirable. With intention to exploit this undesirable property learned by a Neural Machine Translation system we design an attack called: Invariance-based targeted attack. This attack introduces multiple changes(replacement of words) to the original input sentence, keeping the translation unchanged. In-order to facilitate the explanation of the design of the attack we introduce two methods: (i) Min-Grad method: To identify the position where a replacement of the word makes the least change in the translation, and (ii) Soft-Attn method: To search for a new word to replace, given a list of choices.

The initial part of the report explain the preliminary explorations we did in-order to get some insights on how to do the problem formulation. These experiments are run on LSTM based models with single replacement policy. Using the learning from the first part we extend the experiments to Transformer and BLSTM based models, which are considered as the state-of-the-art systems for machine translation.

Introduction

Language translation is a problem of translating sentences in one language to another. A person who has sufficient fluency on the use of two languages can understand what the speaker in the first language is trying to convey and at a later stage, present that idea in a different language. Here in the process of translation, the translator first generates an intermediate representation in their mind and then converts the ideas into a sentence in a different language. With the advent of machine learning, researchers have put lots of efforts to design systems which can do automatic learning and translation. These sequence to sequence machine learning tasks, to translate a sentence from language to another is named as Machine Translation.

The primitive rule-based system has inherent problems with scaling up. Since the rules have to be identified and added to the system with the help of experts, it becomes very hard to look at all edge case rules. Given the availability of large dataset of pairs of sentences which are translations of each other, statistical inference can be extracted in order to make a translation system. This is called Statistical Machine Translation. Let e be the source sentence and f be the target sentence. Given e we need to find a translation f having the highest confidence, ie, $argmax_f(P(f|e))$. This translation system can be re-written in a different form using Bayesian rule as $argmax_f(P(e|f) * P(e))$ where $P(e|f)$ is the alignment model where the word to word mapping from source to target is modeled and $P(e)$ is the language model for the source language.

The application of Neural Networks to solve problems in the image domain showed the magical power of Neural networks to learn from large datasets and also generalize well to predict on unseen data. This prompted researchers to apply Neural Networks in Machine Translation as well. Neural Machine Translation(NMT) is a sequence to sequence task which makes use of Neural Networks for machine learning to translate sentences from one language to another.

With the rise of the Internet and digital content there is huge pressure on international business to localize content in-order to gain optimum online presence. The same argument is applicable for the social networks or instant messaging applications which could allow users to interact in their native language and let the machine translators do the rest of the work. Military and other government agencies also rely on automatically translating languages in-order to monitor contents which are in foreign languages. These are just a small subset of the application of Machine translation. Due the criticality of the applications, correctness and robustness of these systems should also be analyzed with equal importance.

Usual Neural Machine translation systems are encoder-decoder based models. One Neural network acts as the encoder which, takes the input sentence in the source language and encodes it into a

set of vectors called context vectors which contains all the information about the input sentence. A second Neural network will act like the decoder which takes the previously obtained context vector as input and generates the translation in the target language as output. (B)LSTMs were the building blocks and also the baseline for the encoders and decoders until transformer based models were introduced.

Given a sentence and if we are allowed to make some changes to the sentence like a replacement, addition, removal of an (unrelated)word, then a human translator will be able to understand these obvious changes and make a translation reflecting these changes. On the character level, a sentence with and without spelling mistakes gets translated into the same sentence in the target language by a human translator. Humans can easily adapt to the variance at word level and character level. One interesting point to ponder upon is whether a Neural Machine Translation system also has these properties. In this work, we corrupt input sentence, i.e. do multiple replacements of words and are able to demonstrate that the encoder-decoder based machine translation systems are invariant to word level changes.

We will talk about an example from machine translation to make the idea more clear. Consider a surveillance system which translates all the content into a specific language and then make decisions on whether to block the content or not. An attacker is a person who wants to put up some undesirable content online. Let's assume for the time being, the attacker can come up with a method to change the "good" input sentence such that, the "important" information that the attacker wanted, is introduced into the input sentence by replacing the words in the source sentence. The attacker will be successful in putting up this modified content online if its translation will remain "good", i.e. unchanged wrt to the original translation. Here since the translation model was not able to keep track of the changes made in the source side, the attacker was able to introduce unwanted content online even in the presence of a surveillance system.

A more important outcome of this study is to get some insights and look at the drawbacks of the state-of-the-art models. In this era where researchers are investigating on how to explain the inner workings of a machine learning model, this type of analysis will surely contribute to that effort.

We have experimented the attack strategies proposed on multiple types of models for neural machine translation: (i) LSTM based encoder-decoder model, and (ii) BLSTM based models (iii) Transformer based model. Even when the LSTM based translation model is simple, the results will give a good insight into the workings of LSTMs. Transformer-based models are considered to be the current state-of-the-art for machine translation. This model requires huge computation power and large dataset to be trained properly. In the latter part of this dissertation, we also extend the proposed attacks to transformer based models.

1.1 Preliminaries

Before diving deeper into the problem and the solutions, definition of some important terms needs to be understood as they will be extensively used throughout this report.

1.1.1 Recurrent Neural Networks

Classic Neural Networks when used for inputs in the sequential form, there is a chance for the number of parameters to explode or it may not even be possible to decide the size of parameters beforehand. This is where we make use of Recurrent Neural Networks(RNN). An RNN can incrementally process inputs at each time steps separately and the output is an encoding for the whole sequence. Since the parameters used at each time step are shared, the explosion of the number of parameters can be avoided. Each input x of length n is a sequence represented as $(x_1, x_2, ..., x_{n-1}, x_n)$. The RNN at each time step t calculates the hidden representation called hidden state h_t as

$$h_t = F(x_t, h_{t-1}) \tag{1.1}$$

The output o_t for each time step is defined in terms of h_i as

$$o_t = G(h_t) \tag{1.2}$$

In the above equations, $F()$ is an RNN Cell and $G()$ is a Feed-Forward neural network.

1.1.2 Long Short Term Memory Networks

RNN inherently has issues with exploding gradients and vanishing gradients. Long Short Term Memory is one way to address these issues. LSTM was introduced by Hochreiter et al. [1] in 1997. Along with the hidden state defined for RNN, LSTM keeps track of a cell state, which acts as a bypass for the gradients to pass easily. Thus, the cell state helps the LSTM to keep track of long term dependencies. LSTM consists of modules called gates which are: *Input gate*, *Forget gate* and *Output gate*.

$$\begin{aligned} f_t &= F(x_t, h_{t-1}) \\ i_t &= I(x_t, h_{t-1}) \\ o_t &= O(x_t, h_{t-1}) \\ c_t &= f_t \circ c_{t-1} + i_t \circ C(x_t, h_{t-1}) \\ h_t &= o_t \circ H(c_t) \end{aligned} \tag{1.3}$$

where f_t, i_t and o_t are outputs from forget, input and output gates respectively, c_t is the cell state and h_t is the hidden state.

1.1.3 Neural Machine Translation

NMT is one of the fastest growing areas in machine learning. It was introduced by Kalchbrenner et al. [2], Sutskever et al. [3] and Cho et al. [4]. Most NMT systems have an encoder-decoder architecture. The encoder part reads the input and represents the meaning in a fixed(or variable) sized vector which is called the context vector. Then the decoder takes this context vector and generates the translation to a different language. Both the encoder and decoder are neural networks which will be trained jointly in-order to maximize the probability of translation given a source sentence.

Let the source x be the source sentence of length n. x_i's are the words at position i in the sentence x. Similarly y be the target sentence(translation of x) of length m and y_j's be the words in y. The encoder generates a hidden state $h_i = ENC(x_i, h_{i-1})$ for each word x_i and previous time step hidden state h_{i-1}. The context vector mentioned above is a function of all the hidden states from the encoder. But since any length input will be encoded into a fixed length context vector by the encoder, there is a bottleneck in this method for obtaining context vectors. Bahdanau et al.[5] introduced a method to jointly align the source and target along with learning how to do the translation. For the translation model to learn the alignment, the idea of having multiple context vectors for each new time step of the encoder was proposed.

$$c_t = CV(o_{t-1}, h_{t-1}) \tag{1.4}$$

where o_{t-1} is the output from decoder at the previous time step. The output is calculated as a function of the previous output from the decoder and the weighted average on the encoder output states. The weights are called soft attention on the input words which gives the alignment of the source with the target sentence. Figure 1.1 shows a high level diagram of how a Neural Machine Translation system works.

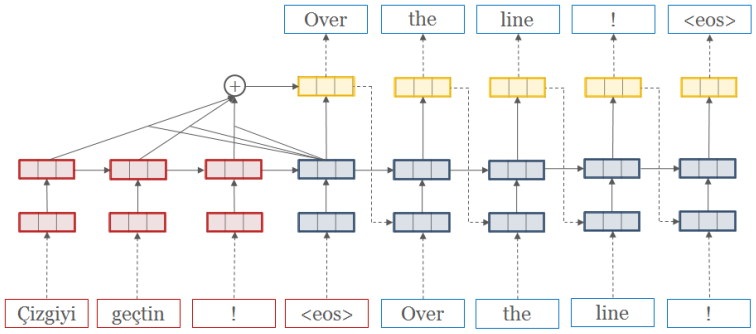

Figure 1.1: Architecture for the translation model in OpenNMT.

1.1.4 Transformers

Vaswani et al. in their paper named "Attention Is All You Need" [6] introduced a new and simple network architecture, the Transformer, based solely on attention mechanisms, dispensing with

recurrence and convolutions entirely. They show that transformer give results superior in quality while being more parallelizable and requiring significantly less time to train. Multiple kinds of attention mechanisms are used in transformer. Self attention will be used independently by encoder and decoder sides to do attention on the input sentence or the generated sentences respectively. Encoder-Decoder attention is a combined attention mechanism which takes care of the input-output sentence alignments. Figure 1.2 shows the architecture for a Transformer model. Bidirectional Encoder Representations from Transformers(BERT) [7] which is derived from Transformers is one of the most hot-topics in NLP in recent times. It is a pre-trained language model, which can be fine-tuned solve other tasks.

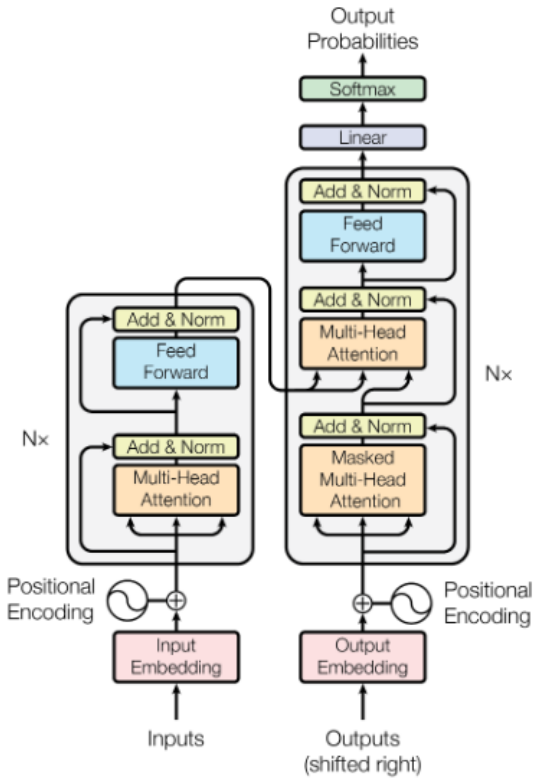

Figure 1.2: Transformer model as given in the paper: Attention is all you need [6].

1.1.5 Adversarial examples

Inputs to a model that are designed to lead the model to generate undesirable or incorrect predictions with very high confidence are termed as adversarial examples. There has been a significant amount of research for designing adversarial attacks for several computer vision tasks (Kurakin et al. [8]; Xu et al. [9]; Xie et al. [10]). All these attacks, in one way or the other, use the gradient of loss with respect to the image for crafting adversarial examples.

Several research has been done on fooling text classifier (Ebrahimi et al. [11]), question answering (Jia and Liang [12]; Feng et al. [13]) and dialogue generation systems (He et al. [14]). Feng et al.

[13] show that question answering models predict the same answer even when most of the words are removed from the question.

Adversarial attacks can be classified in many ways. When the attacker can choose the final output, its called *targeted attack* If the intention of the attacker is just to make the prediction incorrect with very high probability, its called *untargeted attack*. Model weights, loss function, output probability are together called parameters for a model. A Black-Box attack does not make use of these parameters. Whereas the more powerful method White-Box method uses few or all the parameters to do an attack on the model. In general, adversarial examples helps us learn more about how a model would respond to the various regions of input space, especially where the model performs poorly which can aid in understanding and improving the model.

1.1.6 Adversarial attacks in image domain

The results by Szegedy et al. [15] showed that it is possible to fool Deep Neural Networks(to misclassify an image) trained on large datasets such as Imagenet [16] and CIFAR [17] which are having high accuracy scores, by adding hardly perceptible perturbations. The authors show that the network learns some uninterpretable solutions that could have counter-intuitive properties. Nguyen et al. [18] were successful in generating adversarial examples which are totally unrecognizable to human eyes but DNNs believe with near certainty are familiar objects. Authors rely on genetic algorithms and gradient ascent methods to generate the adversarial examples. Previous attempts for explaining the existence of adversarial examples were revolved around the ideas of overfitting and non-linearity of the models. Goodfellow et al. [19] shows and explains this concept by hypothesizing that all the Deep Neural Networks have inherent linearity property. They also came up with a method to make use of adversarial examples during training, called adversarial training. The adversarial training acts like a regularizer which makes the model more robust to adversarial attacks.

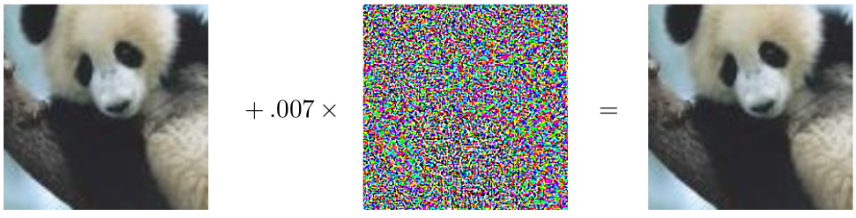

Figure 1.3: Example of adversarial attack on images from Goodfellow et al. [19].

In the Figure 1.3 the initial classification of the image to the class *panda* with 57% confidence is changed to class *gibbon* with 99% confidence by adding an unperceivable noise which is showed in the middle.

1.2 Previous works

In the case of the image domain, subtle changes on images can be made by adding noise to the multi-dimensional array representation of the image. This addition of noise will be imperceptible to humans but the machine learning model will be able to catch the change. In the case of NLP, the input is discrete ie. they are a sequence of words. The word vectors are dense vector representations for a word. Even when the words are encoded using word vectors, they inherently have a discrete nature. The whole set of word vectors are just finite points sampled from an R^d space where d is the size of the vectors. Thus, the trivial operation of adding noise to a word vector cannot guarantee that the newly generated vector is actually valid or not. Unlike images, changing the input in the text is not an operation in continuous space. But selection of a different word from the vocabulary set in-order to corrupt the input makes it a discrete space problem.

In the work by Ebrahimi et al. [11], adversarial examples are generated to fool character-level neural classifiers using a method named as HotFlip. Authors make use of atomic edit operations to flip very few characters in order to greatly reduce the accuracy of the model. The gradient values wrt the loss value on the on-hot-input vectors are used to identify what flip should be made. Since the HotFlip method is very fast in generating adversarial examples, this work also attempts doing adversarial training [19] which makes the final model more robust to attacks.

Let us define the framework for the HotFlip method. Let x be the input text of length L where j^{th} character of i^{th} word in x is represented as $x_{ij} \in \{0,1\}^{|V|}$ where V is the set of alphabets. The character sequence x is represented as $[(x_{11}, x_{12}, ..., x_{1n}); ((x_{21}, x_{22}, ..., x_{2n}); ...((x_{m1}, x_{m2}, ..., x_{mn})]$. Let the log-loss for the model be $J(x, y)$ where x is the input sentence and y as the target. Let the flip operation of character (a $\rightarrow$ b) of j^{th} character of i^{th} word be represented as:

$$\overrightarrow{v_{ijb}} = (\overrightarrow{0}, ..., ; (\overrightarrow{0}, ...(\overrightarrow{0}, ..., \overrightarrow{-1}, \overrightarrow{0}, ..., \overrightarrow{0}, \overrightarrow{1}, ..., \overrightarrow{0})_j, ... \overrightarrow{0})_i; ..., \overrightarrow{0}) \tag{1.5}$$

where -1 and 1 are in the corresponding positions of a^{th} and b^{th} characters of the alphabet and $x_{ij}^{(a)} = 1$. The first order approximation along the vector $\overrightarrow{v_{ijb}}$ can be obtained as:

$$\nabla_{\overrightarrow{v_{ijb}}} J(x, y) = \nabla_x J(x, y)^T \cdot \overrightarrow{v_{ijb}} = \frac{\partial J^{(b)}}{\partial x_{ij}} - \frac{\partial J^{(a)}}{\partial x_{ij}} \tag{1.6}$$

Using the derivative as a surrogate loss and maximizing the above equation we can estimate the best character change. The authors try to make as little change to the input text so that the prediction changes. Here minimum changes in spelling can be considered as a noise. This work also demonstrates HotFlip method based attacks on word level models also.

In a second paper titled "On Adversarial Examples for Character-Level Neural Machine Translation" [20], Ebrahimi et al. extended the HotFlip method to character-level machine translation tasks. Two types of attacks are devised by the authors: (i) Controlled attack where one specific word is removed from the translation and (ii) Targeted attack where the translation of a word is muted as well as a new word is introduced into the translation. All the experiments are done using TED talks parallel corpus prepared by IWSLT 2016.

In the work by He et al. [14], tries to answer a very critical question of whether there exists some input sequence that will cause a well-trained discrete-space neural network sequence-to-sequence

(seq2seq) model to generate egregious outputs. The authors rely on empirical approaches and designs a discrete optimization algorithm to solve this problem. In this work, the authors call an input sequence that causes the model to generate some target (egregious) output sequence a trigger input. Different from adversarial examples in the literature of adversarial attacks, a trigger input is not required to be close to an existing input in the data, rather, they care more about the existence of such inputs. Given a target sentence y of length m, and a trained seq2seq model, we aim to find a trigger input sequence x, which is a sequence of one-hot vectors x_t of length n, which minimizes the negative log-likelihood (NLL) that the model will generate y. The formulation of the objective function L(x;y) is given below:

$$L(x;y) = -\frac{1}{m}\sum_{t=1}^{m} logP_{seq2seq}(y_t|y_{<t},x) + \lambda_{in}R(x) \tag{1.7}$$

The regularization term $R(x)$ is the Language model score of x. Its represented as:

$$R(x) = -\frac{1}{n}\sum_{t=1}^{n} logP_{LM}(x_t|x_{<t}) \tag{1.8}$$

Using Gibbs sampling method, we could obtain a sequence of observations from a specified multivariate probability distribution, especially when direct sampling is difficult. In the above algorithm, the sequence of observations is the sentence and since the loss function $L(x;y)$ is the negative log-likelihood (NLL) that the model will generate y on giving the input as x, it acts as a heuristic for $P(y|x)$. Now the problems boils down to sampling words from the vocabulary such that $L(x;y)$ reduces. This is the basis for the iterative discrete optimization algorithm proposed in this work.

Methodology

This section explains in more detail, the approaches taken to solve the problem. We designed some preliminary experiments to get insights on how to formally design an algorithm, to do an invariance based attack on machine translation system. We will look more closely into these explorations in Section 2.1, as well as how the final design was derived. In Sections 2.1.1 and 2.1.2 we will introduce Min-Grad and Soft-Attn methods. It will act as the building blocks for the solutions proposed in the Section 2.2. The evaluation method is elaborated in the last Section 2.3.

2.1 Preliminary Exploration

When we see any encoder-decoder based Neural Machine Translation systems as a White-Box model, we get the privilege of accessing many levels of information. This includes the weight parameters, loss function and its outputs and class probability values. These types of attacks are called White-Box attacks. The complementary form of attack is called Black-Box attack where there is access only to input sentence and the output translation. Due to the fact that we have more information about the model in a White-Box setting, corresponding attacks should be more powerful.

Let input sentence s be $s = (s_1, s_2, ...s_n)$ where n is the input sentence length and also x_i represent the one-hot encoding of each word s_i. Similarly t denote the be the translated sentence of length m as $t = (t_1, t_2, ..., t_m)$. A translation model $MODEL$, tries to reduce the negative log likelihood loss(L_{nll}) which is defined as follows

$$L_{nll}(s, t) = -\sum_{i=1}^{m} log(MODEL(t_i|t_{<i}, s)) \tag{2.1}$$

where $MODEL(t_i|t_{<i}, s)$ denotes the probability assigned to the word t_i by the NMT model. Now, given an input sentence, we needed to define how to change it such that the L_{nll} is minimized.

For this warm-up experiment, we setup an LSTM based translation system. Then given a position r, we decided to replace the word in the input sentence with another word which is not in the neighborhood of the original one. If a synonymous word replacement identified, there is a possibility of getting a similar translation for the changed sentence as well. This is not a desired behaviour. Thus, we need a new word which should be as different as possible from the original one.

A neighborhood function $N(w, V)$, computes the words from the vocabulary(V) that are close to the given word w according to the following conditions:

1. $Cossim(w, z) >= threshold$
   where $Cossim$ is the cosine similarity between the word embedding of words w and $z \in V$.
2. A lookup dictionary is made to include numbers which are in word form and numeral form. This mapping brings in a similarity constraint on different forms of the same number. As an example, 1 would be mapped to 1.00 and *one* and vice versa.
3. We are also keeping an additional hand written constraint for articles a and an to be in the neighborhood of the number 1.

We needed a method to identify a vulnerable position, such that any replacement at that position will affect L_{nll} the least. The second requirement is to identify a word for replacement from the vocabulary V. For these tasks we introduce two methods in the following sub-sections.

2.1.1 Min-Grad method

Systematically identifying a position for the attack is the initial step. Alternative to taking an input from the user, we propose Min-Grad method to select the position to attack automatically, which makes the attack end to end. Let $e = (e_1, e_2, ..., en)$ denote the word embedding for each word s_i from the input sentence. This is given as input to the LSTM. Let L_{attack} denote the loss function used by the attacking model. We will elaborate more on this difference is loss function for different attack models in later sections. The gradient wrt the word embedding e can be calculated as:

$$grad_i = \nabla_{e_i} L_{attack} \tag{2.2}$$

Gradient of a parameter wrt a loss function gives the sensitivity of the loss function to changes in the parameter's value. Thus, the norm of the gradient of word embedding at each position in a sentence could give a quantitative measure of this sensitivity wrt the loss function. Choosing a position with the smallest norm value gives us a guarantee that the change in loss will be minimum whether its in positive or in negative direction, given the fact that we can make a replacement. Thus, calculation of the position p which has the least sensitivity wrt the loss function can be done as:

$$p = argmin_i(\|grad_i\|) \tag{2.3}$$

2.1.2 Soft-Attn method

For a given position in the sentence, let the word be w. We define a probability distribution $p = (p_1, p_2, ...p_{|V|})$ over all words in the vocabulary. For all the words in $N(w, V)$, we define p_j to be zero. For all the other words we assign uniform probabilities. At the start of each iteration we recalculate the weighted sum of all the word embedding using the probability distribution over

words p and this weighted sum is used instead of the initial w. Using the given loss function L_{attack} as the objective function we update the values for p. Note that we update only those p_i's which are initially non-zero.

We introduce this technique to identify a new word for replacement. The vector of the word to be replaced is represented as weighted average of all the possible word vectors such that sum of the weights is one. Once the new sentence embedding is calculated, use for training the weights as the only parameter, keeping all the other parameters of the model frozen. The weight values after convergence would give the importance in terms of replacement for each word at that position. We select a word such that the effect on the output is minimized.

The Invariance-based attack by doing only single replacement is formalized as given below. We name this attack as Invariance-Attack-Min as it is a minimal version of the actual algorithm.

Algorithm 1 Invariance based attack with single replacement

1: **procedure** INVARIANCE-ATTACK-MIN(s, t, i, V_{pruned}) ▷ s: source; t: target; i: position
2: INITIALIZE zero vector p, of length $|V|$
3: **for** each word index i in V_{pruned} **do**
4: $p_i = \frac{1}{|V_{pruned}|}$
5: INITIALIZE Dictionary COUNT with all initial values to be 0
6: INITIALIZE Sentence embedding for s as e
7: **for** each iteration j from 1 to MAX_ITERATION **do**
8: $e_i = p * W$ ▷ *: matrix multiplication; W: word vectors
9: $loss = L_{attack}(e, t)$
10: Update p by optimizing $loss$ value
11: m_prob = max(p)
12: m_index = argmax(p) ▷ index of the max probability value
13: **for** each wi in V_{pruned} **do**
14: **if** $wi \neq m\_index$ **then**
15: COUNT[wi] = 0
16: **if** $m\_prob > THRESHOLD$ **then**
17: COUNT[m_index] += 1
18: **if** $COUNT[m\_index] > MAX\_COUNT$ **then**
19: **break**
20: **else**
21: COUNT[m_index] = 0
22: **return** (max_index, loss)

In Algorithm 1, we represent the word to be replaced as a weighted average of all the words in the pruned vocabulary V_{pruned}. In experiments with single replacements, $V_{pruned} = V/N(w, V)$, represents the set difference of the complete vocabulary and the neighborhood of the word w at position i. The weight assigned to each word is given as p. Initially we start with equal weights. After the modified sentence embedding is obtained in Step 8, the optimizer optimizes the loss function L_{attack} i.e. basically L_{nll} in case of Machine Translation. After the algorithm satisfies

the termination condition in Step 14, we select the word with the maximum weight-value for replacement. The algorithm returns the index of the selected word(max_index) in the Vocabulary and the final loss value($loss$) obtained.

2.2 Invariance-based Targeted Attack

With the preliminary settings given in the previous section, now we are in a position to design a new attacking strategy for the state-of-the-art models: BLSTM and Transformer. Let sentence s_2 be generated from sentence s_1 after replacing the word at position i. Let the reference translation be t. If $L_{attack}(s_1) < L_{attack}(s_2)$, it is fair to say that the actual translation for s_1 will be having higher probability to be more similar to the target translation t that s_2. The first replacement will be similar to *Invariance-Attack-Min* and in the subsequent iterations replacements are made only if that operation reduces the loss value. This constrain in the algorithm will guarantee a higher success rate than single replacements.

Algorithm 2 Fully targeted attack.

1: **procedure** INVARIANCE-ATTACK(s, t) ▷ s: source; t: target
2: INITIALIZE $list_{rep} = [\,]$
3: s‘ = s
4: $V_{pruned} = V/s$
5: min_loss = MAX_VALUE
6: $l_{org} = L_{attack}(s, t)$
7: flag = **false**
8: **for** iter from 1 to MAX_OUTER_LOOP **do**
9: **for** position i in GET_POSITION_ORDER(n) **do** ▷ n = length of sentence
10: (index, loss) = **INVARIANCE-ATTACK-MIN**(s’, t, i, V_{pruned})
11: l = $L_{attack}(s‘, t)$
12: **if** $i \in list_{rep}$ and $loss < l$ **then**
13: min_loss = max(loss, l_{org})
14: $s'_i = V[index]$
15: flag = **true**
16: **if** $i \notin list_{rep}$ and $loss < min\_loss$ **then**
17: min_loss = max(loss, l_{org})
18: $s'_i = V[index]$
19: $flag$ = **true**
20: APPEND i to $list_{rep}$
21: **if** flag = **false then**
22: **break**
23: **return**(s’) ▷ return the modified sentence

By keeping these key insights in mind, we design an Invariance-based targeted attack. It will be an iterative algorithm, which will repeatedly pass through all the positions in the input sentence in some specific order and perform *Invariance-Attack-Min* 1. The order for looking at the input

sentence is chosen using either Min-Grad method at the start of each iteration or at random. Both selections are done without replacement. This idea is formally presented as Algorithm 2.

Both the Transformer and BLSTM based models uses a shared vocabulary between the encoder and decoder. Let V_{shared} represent this shared vocabulary. Let V_{train} represent all the unique words from the source language training set. For experiments in this section, we make use of the vocabulary of words given by $V = V_{shared} \cap V_{train}$. We do not want words from the original sentence to be sleeted for replacement. To avoid this we remove words in the source sentence from vocabulary V (line #4). The vocabulary size($|V_{shared}|$) of the dataset used for training is about 30K (Table 3.5) is not a very large number. The number of synonyms for a particular word in the vocabulary would be very less, which implies that the probability of generating a paraphrased sentence having the same meaning as the original sentence is negligible. If a paraphrase with reordered words is generated then the BLEU score can capture it. This motivated us to do multiple replacements on the input sentence which will reduce the probability that synonyms are getting replaced. Also to use BLEU score and mean, median, minimum, maximum of number of replacements as evaluation metrics.

2.3 Evaluation Methods

Even when a sentence does not come from the input data distribution, there is a tendency for the model to give a translation similar to the target sentence distribution. In order to avoid this issue, we introduce a way for comparative evaluation. For this, we make use of two models with same source side language ($language_s$), different target languages ($language_{t1}, language_{t2}$). The dataset which the two models were trained on have similar distributions. We are interested in the first target language $language_{t1}$ and use the second target language $language_{t2}$ as a proxy to measure the effectiveness of the proposed method.

If there is some bias present in the dataset, both the models should learn it since they are of the same type and the datasets are also similar. Given the case that both the models had learned similar bias from the dataset, when adversarial sentence is given as input to both the models, the translations would be very close to their corresponding reference translations. Following statements are true given that both the models learned to translate the adversarial input to a sentence very close to the reference translation:

1. The BLEU score between the original sentences and the adversarial sentences should have a low value since we are making multiple replacements.
2. The BLEU score between original and adversarial translations using $language_{t1}$ should be high. If we have high success rates, number of adversarial translations same as the original translation would be high.
3. The BLEU score between original and adversarial translations using $language_{t2}$ should be high due to our assumption.

For our attack to be successful, we need the BLEU score using $language_{t2}$ to be very low. This will show that adversarial examples obtained from the proposed method are indeed successful in attacking the model rather than capturing some biasness in the dataset.

Implementation and Dataset

3.1 Dataset

For the work of this dissertation two datasets are used. For training and testing LSTM based models, Multi30K dataset [21, 22, 23] was used. To train and test the Transformer based models we make use of TED talk dataset [24]. Details about individual datasets are described below.

Multi30K dataset which were originally published for multi-model machine translation task. The task of learning from multiple types of inputs(in this case text and corresponding images) and predicting the translation is called multi-model machine translation. We use three pairs of languages from the dataset namely English-German(en-de), English-French(en-fr) and English-Czech(en-cs). All the languages has 30k sentences each, out of which 29k is the training set and rest 1k is kept as a validation set. Table 3.2 shows the vocabulary sizes for different datasets.

Lang	Source	Target
en-de	a man in an orange hat starring at something .	ein mann mit einem orangefarbenen hut , der etwas anstarrt .
	people are fixing the roof of a house .	leute reparieren das dach eines hauses .
en-fr	professional baseball players during the all star game watch an opponent at bat .	des joueurs de base-ball professionnels lors du all star game regardent un adversaire à la batte .
	two cars are driving on a racetrack .	deux voitures roulent sur un circuit .
en-cs	two people riding bikes through a mountainous region .	dva lidé jedou na kolech přes horskou krajinu .
	women , wearing traditional clothing , are reenacting native life .	ženy v národních krojích předvádějí domorodý život .

Table 3.1: Examples from Multi30k dataset.

Language	Vocab size
English	10212
French	11223
Czech	22400
German	18726

Table 3.2: Vocabulary sizes for Multi30k dataset.

Lang. Pair	BLEU
en-fr	44.69
en-cs	27.61
en-de	34.45

Table 3.3: BLEU score of LSTM models.

Lang	Source	Target
en-de	While that might be a somewhat morbid thought , I think it has some really profound implications that are worth exploring .	Das scheint ein morbider Gedanke zu sein , doch ich finde , er liefert einige ziemlich profunde Implikationen , die eine nähere Betrachtung rechtfertigen .
	We just could not comprehend how this had happened .	Wir konnten uns das einfach nicht vorstellen .
	There were literally thousands of patents on human genes .	Es gab buchstäblich Tausende Patente auf das menschliche Genom .
en-fr	(Applause) This intersection had been bland and anonymous .	(Applaudissements) Cette intersection était fade et anonyme .
	But that costs millions of dollars .	Mais cela coûte des milliards de dollars .
	And we approach this problem by considering another curious syndrome called phantom limb .	Et on approche ce problème en considérant un autre syndrome curieux , appelé le membre fantôme .

Table 3.4: Examples from TED dataset.

Language	Vocab size
english-german	31909
english-french	31758

Table 3.5: Vocabulary sizes for TED dataset.

Lang. Pair	BLEU
en-fr	43.15
en-de	29.27

Table 3.6: BLEU score of Transformer models.

Lang. Pair	BLEU
en-fr	39.32
en-de	26.33

Table 3.7: BLEU score of BLSTM models.

3.2 Implementation details

The PyTorch implementation of OpenNMT [25] from their Github repository[1] is used to train the model(Refer Table 3.3 for evaluation of the trained model on the validation set). This work deals with two types of models as described below:

1. LSTM based model: The trained model is an encoder-decoder with attention. Both encoder and decoder are stacked LSTMs with two layers. LogSoftmax as the final activation layer for the decoder, Negative Log Likelihood as the loss function and ADAM [26] as the optimizer were used for training.

2. Transformer based model: The model uses 6 layers with 8 number of multi-heads. Both RNN hidden size and word vector size is set as 512. The feed forward layer in the transformer has a size of 2048. ADAM [26] is used as the optimizer and Negative Log Likelihood is set as the loss function.

Framework for attack which is written as wrapper on OpenNMT, includes the function to calcu-

[1] https://github.com/OpenNMT/OpenNMT-py

late position using min-grad method and also multiple attack strategies including SoftMax based attack using only encoder/encoder-decoder, Brute force attacks, Fully targeted attack using Brute Force. For the proposed SoftMax based attack, termination condition is true if the number of iterations reach maximum_iteration or the maximum weight value for one of the weights reaches maximum_threshold. Above mentioned is a very flexible framework. New attacks can be easily added to it. Also since all the attacks runs using GPU its also fast to run multiple experiments. Hidden vector size, maximum_iteration, maximum_threshold are kept as 500, 1000, 0.9 respectively.

As a result of the work from Devendra et al. [27], the authors had published their implementation of transformer based machine translation system. We demonstrate our attack on transformers using this framework[2].

In the Soft-Attn method for updating the probability distribution over the set of words(after filtering) we use gradient descent algorithm. The implementation makes use of an additional hack to do gradient accumulation during all the forward and backward propagation made during training for one sentence. Basically the gradients calculated for each parameter wrt the loss function is not zeroed out after each back-propagation and weights are updated. This addition in the implementation gives a huge push for the algorithm to move towards convergence.

Byte pair Encoding(BPE) [28] method is used to do pre-possessing and post-possessing of input to and output from the BLSTM and Transformer based models. BPE can be used to tackle the issues related to rare words and out-of-vocabulary words. It is based on the intuition that various word classes are translatable via smaller units than the original words. The algorithm splits each word into smaller sub-word units which are more commonly seen depending on previously learned sub-word frequency statistics. For applying BPE we made use of the code written by Rico Sennrich. The code is available in GitHub[3].

To compare the similarity between original and generated input sentences and between predicted translation and any reference translations are done using BLEU score [29]. BLEU score was introduced to reduce human intervention for text evaluation and is based on weighted n-gram precision values. We define a metric *Success-Rate*, which is the percentage of translated adversarial sentences which are same as the original translations.

3.3 System level information

All the experiments were run on *GeForce GTX 1080 Ti* GPU with 11175MiB RAM. CPU used is *Intel(R) Core(TM) i7-8700K CPU @ 3.70GHz*. The machine has 62.9 GB of RAM.

[2] https://github.com/DevSinghSachan/multilingual_nmt
[3] https://github.com/DevSinghSachan/multilingual_nmt/blob/master/bin/apply_bpe

Experiments and Results

This Section will discuss and analyze the experiments performed and their results. In Section 4.1 we discuss the experiments conducted on the unidirectional LSTM based models. For LSTM based models, invariance based attacks using information from only the Encoder and both Encoder and Decoder are discussed. In the Section 4.2 we discuss the Attack performed on BLSTM and transformer based models.

4.1 Results on LSTM based models

For LSTM based models, the attacks we devise can be put into different classes. One is the encoder based method, which makes use of only the encoder side information, i.e. the output from the encoder. Given a position, we identify the word for replacement by minimizing the Mean Square Error(MSE) between the encoder hidden states L_{enc} obtained using original and corrupted sentences. Second attack makes use of the output from the Decoder side as well and tries to select a new word such that the Negative Log Likelihood L_{nll} between the predicted and reference translation is minimized. All the attacks are considered to be successful only if the adversarial translation and original translation remains the same.

Method	**Position (r)**	**en-de**	**en-fr**	**en-cs**
HotFlip	l	10.8	7.3	9.0
	sl	12.0	**17.4**	**16.6**
	tl	**14.0**	16.3	13.7
Soft-Attn	l	26.8	27.8	26.9
	sl	29.4	**37.2**	**34.0**
	tl	**31.2**	37.0	30.9
Brute Force	l	34.6	42.6	35.7
	sl	36.4	43.5	**40.6**
	tl	**38.5**	**44.3**	37.5

Table 4.1: Success rate (in %) for encoder based attack for different methods and positions. l, sl, tl stands for last, second last and third last position respectively. For a method and a language pair, the highest success rate across positions is marked in bold.

We demonstrate all the attacks on English-German(en-de), English-French(en-fr) and English-Czech(en-cs) language pairs. The brute force attack will give a lower bound for all attacks which

uses only encoder based information. Thus, it can act as a good baseline for comparison. Given a position r for replacement brute force attack chooses a word from the vocabulary set V such that L_{enc} is minimized. Table 4.1 shows the success rate of three attacks, namely brute force, Soft-Attn and HotFlip using the Encoder side information alone. Its very explicit that *Soft-Attn* method outperforms HotFlip method across all positions and language pairs. This is a clear indication of the power of *Soft-Attn* method.

Method	**en-de**	**en-fr**	**en-cs**
HotFlip	6.2	4.9	5.2
min-grad + HotFlip	16.2	20.9	16.6
random + Soft-Attn	39.6	43.3	39.1
min-grad + Soft-Attn	**58.6**	**62.2**	**56.7**

Table 4.2: Success rate (in %) for encoder-decoder based attack for different methods and positions. The best results are marked in bold. The proposed *min-grad + Soft-Attn* outperforms HotFlip.

Encoder-Decoder based attack takes advantage of all the model parameters and finds a replacement word such that negative log-likelihood error L_{nll} is minimized. Unlike the encoder based attacks, we find the position r for replacement through min-grad method(2.1.1). Table 4.2 shows the success rate by using different combinations of position identifiers and replacement algorithms. *HotFlip* represents the original word based HotFlip method which uses gradient of loss function wrt the one-hot encoding of the input to identify both position and word to replace together. *min-grad + HotFlip* is a variation from the original HotFlip method where the position for is identified using min-grad. Last two methods showed in the Table 4.2, uses *Soft-Attn* to obtain the word to replace but uses random ordering of positions for the first method, whereas uses min-grad method for the last method.

Comparison with vanilla HotFlip and Min-Grad + HotFlip shows that, identifying the most vulnerable position using a different approach improves the success rate. Under Encoder-Only setting we have already seen that Soft-Attn performs better than HotFlip. From the Table 4.2 we see that the Min-Grad + Soft-Attn performs the best. In-order to reinforce the fact that Min-Grad is successful not by chance, we also make a comparison with random position selection along with Soft-Attn(random + Soft-Attn). This strategy also shows an inferior performance compared to the best strategy with a significant margin in their results.

With all the experiments conducted using LSTM based translation models, we can conclude that by combining Min-Grad and Soft-Attn methods we are able to perform a significantly powerful attack.

de	**src**	a girl plays in a small pool.
	adv-src	a girl plays in a cupcake pool
	pred	ein mädchen spielt in einem kleinen pool.
	src	a woman is playing volleyball.
	adv-src	a woman is gambling volleyball.
	pred	eine frau spielt volleyball.
fr	**src**	a man in a lab coat is looking through a microscope.
	adv-src	a man in a lab coat is looking through a bumper.
	pred	un homme en blouse de laboratoire regarde dans un microscope.
	src	a man in a suit is sitting at a bus stop.
	adv-src	a man in a suit is sitting at a bus buffet.
	pred	un homme en costume est assis à un arrêt de bus.
cs	**src**	the white dog is running in the shallow water.
	adv-src	the white dog is running in the tennis water.
	pred	bílý pes běží v mělké vodě.
	src	three men are cooking in a kitchen.
	adv-src	three men are cooking in body kitchen.
	pred	tři muži vaří v kuchyni.

Table 4.3: Examples from attacks on LSTM with only single replacements.

4.2 Results on BLSTM and Transformer models

For both BLSTM and Transformer based model, across all methods and language pairs Min-Grad + Soft-Attn performs the best. This outcome is similar to what we obtained with the LSTM based models. The number of replacements is higher when using random + Soft-Attn. With BLSTM models it's interesting to note that for both the language pairs, the mean and median number of replacements are around 0.6 and still we have high success rates. In case of Transformer models the mean and median falls down to around 0.4, but still the success rates are high enough. HotFlip method consistently gives a low success rate when compared to Soft-Attn method. The higher margin between the success rates between these methods shows the effectiveness of Min-Grad + Soft-Attn method.

We generate adversarial example for the test set of a particular language pair and one model type. Next step is to transfer this generated adversarial example across all model types and language pairs. In Table 4.5, the order in which the BLEU scores are given is: Original/Adversarial source for $language_s$, Original/Adversarial Translation for $language_{t1}$ with $model_1$, Original/Adversarial Translation for $language_{t2}$ with $model_1$, Original/Adversarial Translation for $language_{t1}$ with $model_2$ and Original/Adversarial Translation for $language_{t2}$ with $model_2$. Here $model_1$ is the translation model being evaluated and we compare $model_1$ with $model_2$ to evaluate effectiveness.

The number of replacements done on the source side can be obtained from Table 4.4. As more changes are made, the new sentence becomes more dissimilar to the original sentence. Thus, we can state that the number of replacements and the BLEU scores on the source side are inversely related. We can see that this pattern is followed across all language pairs and methods.

The BLEU score for the adversarial translation with $language_1$ is high across all the methods and language pairs. This shows that the adversarial translations are all very similar to the reference

Model	Method	Metric	en-de	en-fr
BLSTM	random + HotFlip	*NOR*	0.25/0.21/0.93/0.04	0.22/0.2/0.67/0.03
		Success%	25.4	28.2
	min-grad + HotFlip	*NOR*	0.22/0.18/0.93/0.04	0.20/0.16/0.67/0.03
		Success%	31.8	40.2
	random + Soft-Attn	*NOR*	0.61/0.62/1.0/0.1	0.65/0.69/1.0/0.06
		Success%	61.2	64.6
	min-grad + Soft-Attn	*NOR*	0.58/0.6/1.0/0.07	0.62/0.66/1.0/0.05
		Success%	**67.8**	**70.8**
Transformer	random + HotFlip	*NOR*	0.28/0.26/0.72/0.03	0.25/0.22/0.77/0.04
		Success%	35.0	40.6
	min-grad + HotFlip	*NOR*	0.27/0.25/0.76/0.03	0.23/0.21/0.77/0.03
		Success%	45.0	44.0
	random + Soft-Attn	*NOR*	0.45/0.45/1.0/0.05	0.39/0.38/0.83/0.05
		Success%	50.2	59.0
	min-grad + Soft-Attn	*NOR*	0.43/0.44/0.92/0.05	0.37/0.36/0.92/0.04
		Success%	**61.6**	**64.8**

Table 4.4: Success rate (in %) for fully targeted attack across various methods and models. *NOR* represents the mean/median/min/max of the normalized **N**umber **O**f **R**eplacements across all the successfully attacked sentences in the test set. The highest Success% is given in bold.

Model	Method	BLEU Score	
		en-de	en-fr
Transformer	random + HotFlip	51.0/80.5/47.5/36.4/43.7	55.5/85.2/40.3/52.0/36.2
	min-grad + HotFlip	53.2/83.1/49.1/36.5/44.8	57.9/88.4/42.0/54.4/37.7
	random + Soft-Attn	32.0/84.8/29.7/20.6/27.8	33.6/89.8/21.6/32.4/19.1
	min-grad + Soft-Attn	31.2/88.6/31.1/20.6/27.4	35.4/92.0/23.3/34.3/20.3
BLSTM	random + HotFlip	57.1/71.3/48.8/43.9/49.4	60.9/79.6/39.3/58.6/41.7
	min-grad + HotFlip	59.3/75.5/50.4/46.0/52.3	64.0/84.9/41.2/61.8/44.9
	random + Soft-Attn	13.8/87.1/19.2/18.34/21.6	13.0/92.4/10.6/28.3/12.1
	min-grad + Soft-Attn	14.5/89.9/19.7/18.5/22.0	12.7/93.9/9.9/27.2/11.9

Table 4.5: Table shows the BLEU score between the adversarial/original source, translation of adversarial/reference sentences using $model_1/language_1$, $model_1/language_2$, $model_2/language_1$, $model_2/language_2$.

translations. This situation arises even when there is more than 40-50% change in the input. Contrary to the initial assumption about bias in dataset, we are getting low BLEU score for the translations to the $language_2$. This shows that the second model is able to reflect the changes that were made to the source sentence while generating the translations. Also, the BLEU score for the adversarial translation with $language_1$ by $model_2$ is low which shows that the adversarial examples are supposed to have *different* translation in $language_1$.

Tables 4.6 and 4.7 show few successful examples from the attacks on Transformer and BLSTM based translation models. In most cases, it's notable that the new words are not getting translated. This phenomenon is similar to the controlled(mute) attack mentioned in Ebrahimi et al. [20].

en-de	**src**	The first step was to ask a question .
	adv-src	shining first step were 2015 sticking an question .
	pred	Der erste Schritt war , eine Frage zu stellen .
	src	The lists go on and on .
	adv-src	The lists Go aspiration plus wider Hang
	pred	Die Listen gehen weiter und weiter .
	src	(Applause) Thank you . (Applause)
	adv-src	(behalf) Thank you . (Applause Royal
	pred	(Applaus) Danke . (Applaus)
	src	My artistic horizons continue to grow .
	adv-src	ladies artistic horizaffecting continue bounce grow ninth
	pred	Meine knstlerische Horizonte wachsen weiterhin .
	src	The point is , of course , if you want to know about malaria , you ask a malaria expert .
	adv-src	graduated gaining Wind heavily BL planted Isn When You wants shall known across Malaria 180 procedure asking an Malaria expert ;
	pred	Wenn Sie über Malaria wissen wollen , fragen Sie einen Malaria-Experten .
	src	Second , there are many other things that we can do .
	adv-src	Secondly But There re lots others Things viewer we capable implant .
	pred	Zweitens gibt es viele andere Dinge , die wir tun können .
en-fr	**src**	One of every hundred children born worldwide has some kind of heart disease .
	adv-src	One From continuity 100 children born has WALL-E sort ums heart illness Studio
	pred	Une des 100 enfants qui naissent dans le monde a une sorte de maladie cardiaque .
	src	Are governments delivering services to the citizens ?
	adv-src	Be governments delivering services hoo switches citizens rabbit
	pred	Les gouvernements qui procurent les services aux citoyens ?
	src	(Applause) Thank you guys . (Applause)
	adv-src	(eh manipulated Thanks cape Chef etc. (ack M.
	pred	(Applaudissements) Merci . (Applaudissements)
	src	The coil then closes the hole .
	adv-src	La coil joins solo On meets hole Magazine
	pred	La bobine se ferme dans le trou .
	src	So how do you actually go around finding online criminals ?
	adv-src	occasions how now Let KA raft Of finding online criminals R.
	pred	Alors , comment fait-on pour trouver des criminels en ligne ?
	src	'Now the question is : How could we ask them ?
	adv-src	Le La matters isn - how Let inviting pose them Kickstarter
	pred	La question est : comment les poser ?

Table 4.6: Examples from attacks on BLSTM with multiple replacements.

en-de	**src**	The corals are doing better here . Lots of small fish .
	adv-src	Arabia corals larvae doing better here . Lots of small fish transistors
	pred	Die Korallen machen hier besser . Viele kleiner Fische .
	src	Not a single body should remain undiscovered or unidentified .
	adv-src	unaware topic single body should remain undsubmitted covered Within uniunclear fied surely
	pred	Kein einziger Körper sollte unbehandelt oder geklärt bleiben .
	src	On the right side you can see my colleague Soren , who 's actually in the space .
	adv-src	On rank your side persistent fitted unequal my colleague Soren recognizes who strand actually oms along space te
	pred	Auf der rechten Seite sehen Sie meinen Kollegen Soren , der tatsächlich im Raum ist .
	src	Second , there are many other things that we can do .
	adv-src	Second has bunch Laughter outbreak other things WK we can action .
	pred	Zweitens gibt es viele andere Dinge , die wir tun können .
	src	Joker : Whoa.MT : Are you ready for your party piece ?
	adv-src	Joker : beside export contributions vessel outing So Way ready behalf yourself party Q Jersey
	pred	Joker : Sind Sie bereit für Ihr Party ?
	src	So you , you give it everything you got and you wind up on the street .
	adv-src	soils forth gifts du giving printer anything you yeah and pedal picks grab upon The street encountered
	pred	Man gibt ihm alles , was man hat , und man greift auf die Straße .
en-fr	**src**	I ran through the numbers myself .
	adv-src	hits Ça My entitled numbers across .
	pred	J'ai traversé mes chiffres .
	src	And I take it that you appreciate that kind of transformation .
	adv-src	Et I understand activate acute thou enjoying that kind grasp transformation bunch
	pred	Et je comprends que vous appréciez ce genre de transformation .
	src	This is an amazing story and adventure for you to go on .
	adv-src	This isn intensive amazing story et adventure favor you rule 4,000 hadn !
	pred	C'est une histoire incroyable et une aventure pour vous .
	src	Well the reason , I think , essentially is , again , back to the civilization-state .
	adv-src	Well du reason by I turns ahead guided Has degree again stands turn to Stuart civilimmersed arena Stage :
	pred	Eh bien , la raison , je pense , est , encore une fois , vers l' état civilisé .
	src	And unlike all of the other animals , we are obsessed with trying to figure it out .
	adv-src	And unlike all absurd personally other animal Isn we amounts obsessed ya trying 've figured him consists apart
	pred	Et contrairement à tous les autres animaux , nous sommes obsédés par l'essayer de le comprendre .
	src	So again , what we focus on is prevention rather than treatment , at first rate .
	adv-src	brief Again danced closure We focusing Sir des prevention Instead than treatment ready inform recently start .
	pred	Encore une fois , nous nous concentrons sur la prévention plutôt que le traitement , au début .

Table 4.7: Examples from attacks on Transformer with multiple replacements.

Conclusion and Future Works

In the image domain, there have been many techniques introduced to generate adversarial examples from the simple FGSM attack [19] to the very powerful C&W method [30]. And the number of relevant works in the Natural Language Processing domain is relatively very low. Thus, the work done for this dissertation is an effort made to demonstrate the weakness in Neural Machine Translation systems and the methods introduced can be extended for other NLP related tasks as well. Once we understand the inherent weaknesses in these systems it opens up a huge opportunity to make improvements to the system. Analyzing NLP systems for the effectiveness of handling adversarial has a huge importance in this Internet era. This work is just a small step in that direction.

The problem formulation for adversarial attacks on a sequence to sequence task has taken ideas from discrete optimization techniques. This idea is clearly visible in this work as well as other previous works including the works of Ebrahimi et al. [11] and He et al. [14]. A more interesting fact is that the solutions devised are all based on the approximated use of gradient-based methods. The results show us that even the gradient-based methods work really well.

In the previous state-of-the-art method HotFlip [20], authors try to find the most vulnerable(optimal in this case) position and a new word to be replaced, simultaneously. In this work, we showed that the finding solution for both the problems separately would give us more control over the attack. Also, the two new methods proposed, ie. Min-Grad method for finding the optimal position and the Soft-Attn method to find the replacement word shows better results when compared with HotFlip at the word level.

For all the three models on which we did the experiments, on common fact stands out: Even with more than 40-50% number of replacements on the source sentence, there is a high chance that the translation remains the same. Our results also show that the adversarial sentences generated are different from the original source. Even with these changes to syntax and semantics of the original sentence, if the model is able to generate an unaltered translation then we should be seriously thinking about what these models are actually learning.

Due to the inherent discrete nature of the inputs, the gradient-based approaches will have its limitations for its application to generate adversarial examples in NLP domain. This idea motivates us to start thinking about alternative approaches which are non-gradient based optimization techniques for the task. A different perspective of looking at this problem as a complete combinatorial or Black-Box optimization problem could enable us to start exploring techniques like Genetic Algorithms and some Sampling Techniques. In their work to generate adversarial examples Moustafa

et al. [31] devises an attack algorithm that exploits population-based gradient-free optimization via genetic algorithms. The main aim is to minimize the number of modified words between the original and adversarial examples, but only perform modifications which retain semantic similarity with the original and syntactic coherence.

By the time attacking techniques becomes popular, we should start looking at the defense mechanisms. Currently, adversarial training is the most popular method for defense against adversarial attacks on NLP models. Even then most works have reported only minor improvement in robustness even after this.

Compared to the image domain, adversarial examples generation in NLP is still in the infant stage. Let it be types of attacks, defense mechanisms or evaluation of adversarial examples, there is a huge potential for further research.

Bibliography

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

[2] Nal Kalchbrenner and P Blunsom. Recurrent continuous translation models. *EMNLP 2013 - 2013 Conference on Empirical Methods in Natural Language Processing, Proceedings of the Conference*, 3:1700–1709, 01 2013.

[3] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to sequence learning with neural networks. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'14, pages 3104–3112, Cambridge, MA, USA, 2014. MIT Press.

[4] Kyunghyun Cho, Bart van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder–decoder approaches. In *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, pages 103–111, Doha, Qatar, October 2014. Association for Computational Linguistics.

[5] Dzmitry Bahdanau, Kyunghyun Cho, and Y Bengio. Neural machine translation by jointly learning to align and translate. *ArXiv*, 1409, 09 2014.

[6] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, pages 6000–6010, USA, 2017. Curran Associates Inc.

[7] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805, 2018.

[8] Alexey Kurakin, Ian J. Goodfellow, and Samy Bengio. Adversarial examples in the physical world. *CoRR*, abs/1607.02533, 2016.

[9] Xiaojun Xu, Xinyun Chen, Chang Liu, Anna Rohrbach, Trevor Darrell, and Dawn Song. Fooling vision and language models despite localization and attention mechanism. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[10] Cihang Xie, Jianyu Wang, Zhishuai Zhang, Yuyin Zhou, Lingxi Xie, and Alan L. Yuille. Adversarial examples for semantic segmentation and object detection. *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 1378–1387, 2017.

[11] Javid Ebrahimi, Anyi Rao, Daniel Lowd, and Dejing Dou. HotFlip: White-box adversarial examples for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 31–36, Melbourne, Australia, July 2018. Association for Computational Linguistics.

[12] Robin Jia and Percy Liang. Adversarial examples for evaluating reading comprehension systems. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2021–2031. Association for Computational Linguistics, 2017.

[13] Shi Feng, Eric Wallace, Alvin Grissom II, Mohit Iyyer, Pedro Rodriguez, and Jordan Boyd-Graber. Pathologies of neural models make interpretations difficult. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3719–3728. Association for Computational Linguistics, 2018.

[14] Tianxing He and James Glass. Detecting egregious responses in neural sequence-to-sequence models. In *International Conference on Learning Representations*, 2019.

[15] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian J. Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *CoRR*, abs/1312.6199, 2014.

[16] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009.

[17] Alex Krizhevsky. Learning multiple layers of features from tiny images. *University of Toronto*, 05 2012.

[18] Anh Nguyen, Jason Yosinski, and Jeff Clune. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 12 2014.

[19] Ian Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv 1412.6572*, 12 2014.

[20] Javid Ebrahimi, Daniel Lowd, and Dejing Dou. On adversarial examples for character-level neural machine translation. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 653–663, Santa Fe, New Mexico, USA, August 2018. Association for Computational Linguistics.

[21] Desmond Elliott, Stella Frank, Khalil Sima'an, and Lucia Specia. Multi30k: Multilingual english-german image descriptions. In *Proceedings of the 5th Workshop on Vision and Language*, pages 70–74. Association for Computational Linguistics, 2016.

[22] Desmond Elliott, Stella Frank, Loïc Barrault, Fethi Bougares, and Lucia Specia. Findings of the second shared task on multimodal machine translation and multilingual image description. In *Proceedings of the Second Conference on Machine Translation, Volume 2: Shared Task Papers*, pages 215–233, Copenhagen, Denmark, September 2017. Association for Computational Linguistics.

[23] Loïc Barrault, Fethi Bougares, Lucia Specia, Chiraag Lala, Desmond Elliott, and Stella Frank. Findings of the third shared task on multimodal machine translation. In *Proceedings of the Third Conference on Machine Translation: Shared Task Papers*, pages 304–323, 2018.

[24] Ye Qi, Devendra Sachan, Matthieu Felix, Sarguna Padmanabhan, and Graham Neubig. When and why are pre-trained word embeddings useful for neural machine translation? In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 529–535, New Orleans, Louisiana, June 2018. Association for Computational Linguistics.

[25] Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander M. Rush. OpenNMT: Open-source toolkit for neural machine translation. In *Proc. ACL*, 2017.

[26] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *International Conference on Learning Representations*, 12 2014.

[27] Devendra Sachan and Graham Neubig. Parameter sharing methods for multilingual self-attentional translation models. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 261–271, Belgium, Brussels, October 2018. Association for Computational Linguistics.

[28] Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany, August 2016. Association for Computational Linguistics.

[29] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA, July 2002. Association for Computational Linguistics.

[30] Nicholas Carlini and David A. Wagner. Towards evaluating the robustness of neural networks. *CoRR*, abs/1608.04644, 2016.

[31] Moustafa Alzantot, Yash Sharma, Ahmed Elgohary, Bo-Jhang Ho, Mani Srivastava, and Kai-Wei Chang. Generating natural language adversarial examples. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2890–2896, Brussels, Belgium, October-November 2018. Association for Computational Linguistics.