Seq2Sick: Evaluating the Robustness of Sequence-to-Sequence Models with Adversarial Examples

M. Cheng¹, J. Yi², H. Zhang¹, P. Chen³, C. Hsieh¹

¹University of California, Davis ²Tencent AI Lab ³IBM Research

arXiv: 1803.01128
Reviewed by: Bill Zhang
University of Virginia
https://qdata.github.io/deep2Read/
Outline

Introduction
Related Work
Methodology
Experiments
Results
Conclusion
References
Introduction
Basic Premise and Motivation

- There are many attacks for DNNs, but much less for text models
- Attacking a text string is difficult because input space is discrete and output space (if a sequence of words) can be near infinite compared to classification problems
- Targeted attacks are especially difficult because of near infinite output space
- Robustness of seq2seq important because of wide usage in machine translation, text summarization, and speech recognition
Related Work

- Gradient, score, transfer, and decision-based methods for attacking CNN-based models
- FGSM to attack RNN/LSTM-based models
- Reinforcement learning to learn important words to delete in sentiment classification
- Replacing words with typos/synonyms
- Scoring function to find most important words to modify
- Adding misleading sentences for summarization
- GAN to generate examples (only works for untargeted and computationally expensive)
- Most previous methods are based on a greedy search
Methodology

Seq2Seq

- $X = (x_1, x_2, ..., x_N)$ to $Y = (y_1, y_2, ..., y_M)$, where $x_i \in \mathbb{R}^d$ is the embedding vector of each input word
- Each RNN/LSTM cell computes $h_t = f(x_t, h_{t-1})$
- Compute context vector $c = q(h_1, h_2, ..., h_N) = h_N$
- $z_t = g(y_{t-1}, c)$ and $p_t = \text{softmax}(z_t)$ to predict next word
Crafting against adversarial examples is following optimization problem

$$\min_{\delta} L(X + \delta) + \lambda R(\delta)$$

where $R$ is a regularization function to measure magnitude of distortion and $L$ is a loss function

- Common $R$ is $l_2$ loss, but unsuitable for seq2seq
- Focus on 2 attacks: non-overlapping and targeted; disregard untargeted due to triviality of causing only one-word difference
Methodology
Non-Overlapping Attack

- Non-overlapping attack requires every output word in sequence to be different from original output word; If $s = s_1, s_2, \ldots, s_M$ is the original output sequence and $v$ is output vocabulary, then

$$s_t \neq \arg\max_{y \in v} z_t^{(y)} \quad \forall t = 1, 2, \ldots, M$$

$$z_t^{(s_t)} < \max_{y \in v, y \neq s_t} z_t^{(y)} \quad \forall t = 1, 2, \ldots, M$$

- Thus, let loss be

$$L_{\text{non-overlapping}} = \sum_{t=1}^{M} \max \{-\epsilon, z_t^{(s_t)} - \max_{y \neq s_t} \{z_t^{(y)}\}\}$$

where $\epsilon \geq 0$ is the confidence margin parameter (larger values will lead to more confident output)
Methodology
Targeted Keywords Attack

- Targeted keywords requires the output to have all target keywords in output sequence; does not matter what position
- First, define following loss function where $K = k_1, k_2, \ldots, k_{|K|}$ is list of target keywords

\[
L_{targeted} = \sum_{i=1}^{|K|} \min_{t \in [M]} \left\{ \max\{-\epsilon, \max_{y \neq k_i} \{z_t^{(y)}\} - z_t^{(k_i)}\} \right\}
\]

- To avoid competing keywords, apply mask
  \[
m_t(x) = \{\infty, \text{ if } \arg\max_{i \in \{z_t^{(i)}\} \in K; x, \text{ otherwise}}\}
\]
- Final loss function is

\[
L_{targeted} = \sum_{i=1}^{|K|} \min_{t \in [M]} \{m_t(\max\{-\epsilon, \max_{y \neq k_i} \{z_t^{(y)}\} - z_t^{(k_i)}\})\}\]
Methodology
Discrete Input Space

- Naive method is to search for optimal $X + \delta^*$ in continuous space then search for nearest embedding in word-space $\mathbb{W}$; not effective because final solution likely not a feasible word embedding in $\mathbb{W}$ (nearest neighbor could be far away)
- Change optimization function to

$$\min_{\delta} L(X + \delta) + \lambda R(\delta) \text{ s.t. } x_i + \delta_i \in \mathbb{W} \ \forall i = 1, 2, \ldots, N$$

at each step of PGD, project current solution back into $\mathbb{W}$

- Use Group Lasso regularization to enforce group sparsity so that few words in input are changed

$$R(\delta) = \sum_{t=1}^{N} \|\delta_t\|_2$$
Methodology
Gradient Regularization

- Common for adversarial example to be located in region with very few embedding vectors; even closest embedding from PGD can be far away
- Add to loss function to make $X + \delta$ close to word embedding space

$$
\min_{\delta} L(X + \delta) + \lambda_1 R(\delta) + \lambda_2 \sum_{i=1}^{N} \min_{w_j \in W} \{||x_i + \delta_i - w_j||_2\}
$$

s.t. $x_i + \delta_i \in W \ \forall i = 1, 2, ..., N$
Experiments
Datasets, Seq2Seq Models

- DUC2003, DUC2004, Gigaword for text summarization attack, WMT’16 Multimodal Translation for machine translation
- Implement models on OpenNMT-py, specifically a word-level LSTM encoder and word-based attention decoder
- Use hyperparameters suggested by OpenNMT
Results

Text Summarization

- Non-overlapping results: change 2 to 3 words to change 80% of outputs

<table>
<thead>
<tr>
<th>DATASET</th>
<th>SUCCESS RATE</th>
<th>BLEU</th>
<th># CHANGED</th>
</tr>
</thead>
<tbody>
<tr>
<td>GIGAWORD</td>
<td>86.0%</td>
<td>0.828</td>
<td>2.17</td>
</tr>
<tr>
<td>DUC2003</td>
<td>85.2%</td>
<td>0.774</td>
<td>2.90</td>
</tr>
<tr>
<td>DUC2004</td>
<td>84.2%</td>
<td>0.816</td>
<td>2.50</td>
</tr>
</tbody>
</table>
Results

Text Summarization

- Targeted results: very successful with 1 or 2 target keywords; less successful, but still able to find examples for 3 keywords

| DATASET    | $|K|$ | SUCCESS RATE | BLEU  | # CHANGED |
|------------|-----|--------------|-------|-----------|
| GIGAWORD   | 1   | 99.8%        | 0.801 | 2.04      |
|            | 2   | 96.5%        | 0.523 | 4.96      |
|            | 3   | 43.0%        | 0.413 | 8.86      |
| DUC2003    | 1   | 99.6%        | 0.782 | 2.25      |
|            | 2   | 87.6%        | 0.457 | 5.57      |
|            | 3   | 38.3%        | 0.376 | 9.35      |
| DUC2004    | 1   | 99.6%        | 0.773 | 2.21      |
|            | 2   | 87.8%        | 0.421 | 5.1       |
|            | 3   | 37.4%        | 0.340 | 9.3       |
Results

Text Summarization

- Test significance of each component of objective
  - Removing PGD dropped success to 0%, shows importance of projecting back into input vocabulary word embeddings
  - Removing group lasso does not change success significantly, but does change increase of words changed and decrease BLEU score
  - Removing gradient regularization can lower success rate

<table>
<thead>
<tr>
<th>DATASET</th>
<th>METHOD</th>
<th>SUCCESS%</th>
<th>BLEU</th>
<th># CHANGED</th>
</tr>
</thead>
<tbody>
<tr>
<td>GIGAWORD</td>
<td>W/o GL</td>
<td>91.4%</td>
<td>0.166</td>
<td>16.53</td>
</tr>
<tr>
<td></td>
<td>W/o GR</td>
<td>92.8%</td>
<td>0.707</td>
<td>4.96</td>
</tr>
<tr>
<td></td>
<td>ALL</td>
<td>96.5%</td>
<td>0.523</td>
<td>4.96</td>
</tr>
<tr>
<td>DUC2003</td>
<td>W/o GL</td>
<td>95.7%</td>
<td>0.225</td>
<td>15.74</td>
</tr>
<tr>
<td></td>
<td>W/o GR</td>
<td>87.9%</td>
<td>0.457</td>
<td>5.57</td>
</tr>
<tr>
<td></td>
<td>ALL</td>
<td>87.6%</td>
<td>0.457</td>
<td>5.57</td>
</tr>
<tr>
<td>DUC2004</td>
<td>W/o GL</td>
<td>95.0%</td>
<td>0.212</td>
<td>15.60</td>
</tr>
<tr>
<td></td>
<td>W/o GR</td>
<td>87.0%</td>
<td>0.421</td>
<td>5.14</td>
</tr>
<tr>
<td></td>
<td>ALL</td>
<td>87.8%</td>
<td>0.421</td>
<td>5.14</td>
</tr>
</tbody>
</table>
Results
Machine Translation

- Similar to summarization, obtain results for non-overlapping and targeted

<table>
<thead>
<tr>
<th>METHOD</th>
<th>SUCCESS%</th>
<th>BLEU</th>
<th># CHANGED</th>
</tr>
</thead>
<tbody>
<tr>
<td>NON-OVERLAP</td>
<td>89.4%</td>
<td>0.349</td>
<td>3.5</td>
</tr>
<tr>
<td>1-KEYWORD</td>
<td>100.0%</td>
<td>0.705</td>
<td>1.8</td>
</tr>
<tr>
<td>2-KEYWORD</td>
<td>91.0%</td>
<td>0.303</td>
<td>4.0</td>
</tr>
<tr>
<td>3-KEYWORD</td>
<td>69.6%</td>
<td>0.205</td>
<td>5.3</td>
</tr>
</tbody>
</table>
Results

Machine Translation

- Once again, test significance of each component
  - Removing PGD dropped success to 0%
  - Removing group lasso increased success at the cost of words changed and BLEU
  - Removing gradient regularization had small negative impacts on results

<table>
<thead>
<tr>
<th>METHOD</th>
<th>SUCCESS RATE</th>
<th>BLEU</th>
<th># CHANGED</th>
</tr>
</thead>
<tbody>
<tr>
<td>W/O GL</td>
<td>100.0%</td>
<td>0.163</td>
<td>6.4</td>
</tr>
<tr>
<td>W/O GR</td>
<td>91.0%</td>
<td>0.303</td>
<td>4.1</td>
</tr>
<tr>
<td>ALL</td>
<td>91.0%</td>
<td>0.303</td>
<td>4.0</td>
</tr>
</tbody>
</table>
Results

Robustness of Seq2Seq

- Attack methods proposed are effective, as shown by results
- Harder to turn entire seq2seq output into particular sentence (sometimes impossible)
- Easier for human to detect differences in inputs due to discrete input space
- Thus, seq2seq is more robust than DNN models
Conclusion

- Seq2sick is a novel framework capable of producing adversarial examples for seq2seq models
- Use PGD to address issue of discrete input space, group lasso to enforce sparsity of distortion, and gradient regularization to further improve success
- Addresses harder problem than previous frameworks which perform untargeted or classification attacks
- Framework is effective, but also recognize robustness of seq2seq compared to DNN
References