Predicting Adversarial Examples in Language using Graphs

Erik Jones
erjones@stanford.edu

Abstract

Deep neural networks that achieve promising results prove brittle in the face of adversarial examples. These adversarial examples look nearly indistinguishable to standard inputs, but their corresponding prediction can be arbitrarily manipulated. Our focus is on adversarial typos attacking the Stanford Sentiment Treebank, which flip the sentiment of a movie review from positive to negative, or vice-versa. Though seemingly random, adversarial typos often generalize between different models, implying the existence of some underlying vulnerability. We attempt to examine the extent to which this underlying vulnerability is the byproduct of some graph structure. We consider all edit distance one typos from a specific word in the input, test whether or not a fixed number of perturbations of that word flip the model prediction, and examine the extent to which label propagation algorithms are able to predict the rest. Our results indicate that there is sufficient structure in induced graphs within natural language data to predict whether or not an example will fool a non-robust classifier.

1 Introduction

Recently, deep neural networks have proven effective at a variety of prediction tasks in NLP. Recent advances in model architecture and in language modeling, namely the pretrained transformer BERT [3], have achieved state of the art results in a suite of NLP tasks [17]. However, models that ostensibly achieve impressive performance standards degrade catastrophically in the face of *adversarial typos*, which are deliberately chosen to fool a specific model classifier. Identifying and defending against these typos is critical in domains like spam detection and hate-speech moderation, during which potentially devastating content needs to be filtered effectively.

Interestingly, previous work has demonstrated that adversarial examples transfer between different models, which have different parameters and architectures [16]. However, there has not been conclusive evidence for why these transfer. One explanation, presented in [7], suggests that adversarial examples are a byproduct of the data; certain biases allow for features that shouldn't be predictive to be leveraged for great gain. Identifying why adversarial examples appear can help inform how to defend against them, and also potentially improve attacks by reducing the query load to very large, computationally expensive models.

We instead hypothesize that there's some structure to adversarial examples in the input space. In particular, we consider typos of a single word in the input that can potentially change the model prediction. These typos have some underlying structure; some typos are closer in edit distance space than others. We leverage this structure by viewing predicting whether or not an example will be adversarial or not as a label propagation problem. We consider different types of graphs (only insertions, only deletions, only substitutions, only swaps, and all four) and different amounts of initial labels. To get and evaluate true labels, we use a character-embedding based bi-directional LSTM [14] trained on the Stanford Sentiment Treebank [15]. We find that even simple label propagation algorithms have some predictive power, and typos connected by substitutions are especially predictive.

2 Related Work

Adversarial examples have revealed vulnerability in current deep architectures [5]. We discuss current models of attack, current work discussing transferability of adversarial examples, and potential graph algorithms to use.

2.1 Valid Attacks

Adversarial examples. In vision, it is easy to construct adversarial examples that are indistinguishable to a human using L_∞ perturbations; each pixel can be modified by a small amount, but not enough that a human can recognize the difference [13]. In natural language processing, however, it is impossible to make an indistinguishable change since the space of possible inputs is discrete.

The common standard for perturbations in natural language processing is *human recoverability*; a perturbation is human recoverable if some human could look at the perturbed input and discern, with high probability, what the original perturbations. In this work, we will study adversarial typos, which a human should be able to recover.

Existing attack surfaces In an attempt to enforce this human recoverability constraint, several different attack surfaces have been proposed. These attack surfaces primarily leverage three edit distance one operations: single-character insertions, delections, substitutions, and adjacent character swaps. Examples of each are as follows:

- Insertion: tired $\rightarrow$ tiared
- Deletion: tired $\rightarrow$ tred
- Substitution: tired $\rightarrow$ tqred
- Swap: tired $\rightarrow$ tried

Notice, as in the last case, it is acceptible to perturb a word to another word. The first attack surface, introduced by Ebrahimi et al. [4] uses only a fixed number of insertions and deletions per sentence, without bounding the number of modifications made to each word. Another attack surface, proposed by Pruthi et al. [12], allows a single insertion, deletion, substitution (only of keyboard adjacent characters) or adjacent character swap of at most two words in the input sentence, provided none of the modifications change the first and last characters of each word in an attempt to maintain human interpretability. The broadest attack surface to date, submitted anonymously to ICLR [1], allows for the same modifications as Pruthi et al. (including now unconstrained substitutions) to *every* word in the input.

We use the BiLSTM presented in Pruthi et al.[12]. Because this model is not robust, it's sufficient to use a single edit distance one substitution. We don't put the keyboard constraint on substitutions.

2.2 Transferability of Adversarial Examples

Interestingly, adversarial examples seem to transfer between models with different architectures and different parameters. Tramèr et al. shows that the space of adversarial examples that transfer is relatively large in dimensionality, perhaps indicating that the transferability of adversarial examples is a byproduct of overparameterization. Xie et al. shows that one can infer which adversarial examples transfer best, which they use as a more efficient attack than some previous work. Similarly Liu et al. shows that not only do adversarial examples transfer, but they also preserve their label, indicating some stronger systematic bias [10]. However, all of these approaches apply to images; limited work has been done showing transferability is still relevant for NLP.

2.3 Label Propagation Algorithms

Label propagation algorithms have been shown to be effective in a variety of settings. Baluja et al. [2] introduce an *adsorption algorithm*, which constructs a probability over labels for each node, which has been widely used in a variety of applications. Perozzi et al. [11] use random walks to generate more compelling features to use for label propagation, as opposed to simply using the current probability distribution over labels at each node. More recently, Kiph and Welling [9] introduce *graph*

convolutional networks which generate features by convolving over the previous layer's features that correspond to each vertices neighbors. Graph convolutional networks achieved state of the art results in several graph based tasks.

3 Setup

3.1 Graph Formulation

In this work, we consider edit distance one typos: insertions, deletions, substitutions, and adjacent character swaps, so a *single* word in an input, as described in the related work. We can view these edit distance one typos as a graph $G = (V, E)$. For a word w, define $B(w)$ to be the edit distance one typos of w. Defining $V = B(w)$, and:

$$E = \{(p, q) \mid p, q \in B(w),\ p \in B(q)\} \tag{1}$$

Notice that edges can be undirected, since if $p \in B(q)$, then $q \in B(p)$ by the construction of our perturbation space. We consider various weaker definitions of B as well, including just insertions, just deletions, and just substitutions. In the case where we deal with directed edges, which can arise when $x \in B(y)$ but $y \notin B(x)$ (think of deletions for motivation), we convert the directed graph into an undirected graph. Formally, we amend our definition of E as follows:

$$E = \{(p, q) \mid p, q \in B'(w),\ p \in B'(q) \vee q \in B'(p)\} \tag{2}$$

We differentiate B', the set of allowable perturbations when applying a single type of edit distance one alteration, from B so that the vertex set can remain constant, even when we slightly change the perturbation model.

3.2 Label Propagation Algorithms

For the purpose of the milestone, we use a simple label propagation baseline. Consider a vertex v and it's neighbors $N(v) = B(v) \cap B(w)$. For each vertex, let $P(v)$ be the probability that vertex v is labeled 1. We choose some set $L \subseteq V$, and run the biLSTM to label those vertices. For each vertex $v \in V \setminus L$, we initialize $P(v) = 0.5$. At each iteration, we update $P(v)$ as follows:

$$P(v) = \frac{1}{|N(v)|} \sum_{x \in N(v)} P(x) \tag{3}$$

Vertices in L that have been labeled stay fixed, and each iteration proceeds in random order. To label the entire graph, we repeat for either 100 iterations or convergence, whichever comes first.

4 Data and Model

In this section, we discuss the dataset we use and the model we attack.

4.1 Dataset

To evaluate the extent to which graph structure helps identify adversarial examples, we construct generic sentences based on movie reviews. These movie reviews are inputs to a two-class classification task; since SST-2 is a sentiment analysis dataset, each word can be classified as positive or negative.

Notice that each movie review has many words, each of which can be perturbed. Moreover, the structure of the perturbations of each words yields different graph structure. Thus, to test transferability between adversarial examples of the same input, each sentence requires constructing an entirely new graph. Due to computational constraints, the total number of sentences we may consider perturbing is limited. How, then, can we test for *general* patterns between graph structure, and the transferability of adversarial examples.

Graph-class variation. To test more broadly for the ability of graphs to capture structure in the space of adversarial examples, we define many different sets of perturbations $B(w)$ for each word w. Our default includes edit distance one insertions, deletions, substitutions, and character swaps.

However, understanding *which* of these types of perturbations contributes most to the structure we seek to capture, we consider smaller perturbation sets $B(w)$. These sets only allow perturbations based on one of our four possible classes of perturbations. We want to examine how predictive these sparser graphs are.

To examine how predictive these sparser graphs are, for each sentence we try, we construct four graphs: one that defines edges based on insertions, one based on deletions, one based on substitutions, and one that uses all three in addition to swaps. We keep the total set of nodes we consider fixed (everything edit distance one under any perturbation class from the original word,) but only add edges based on the defined perturbation type. Notice that insertions and deletions form directed graphs: for example *"ct"* is a deletion of *"cat"*, but *"cat"* is not a deletion of *"ct"*. To remedy this, we make the entire graph undirected.

Choosing Reviews. Due to computational constraints, it is intractible to construct graphs for all of SST-2, which contains over 60,000 movie reviews. Instead, we choose four movie reviews, and construct five graphs for each review. We then attack one word per review. Specifically, we attack *"fine"* in the sentence *"this movie was fine"*, *"fantastic"* in *"a fantastic movie"*, *"unfocused"* in *"a movie with unfocused acting"*, and *"cumbersome"* in *"what a cumbersome plot"*. This gives us 16 total graphs for our analysis.

We give more detailed graph statistics for attacking *"fine"* in the sentence *"this movie was fine"* The model we attack is trained on SST-2 [15], and the labels are positive and negative. There are a 130 perturbations of fine, and the induced graph contains 1911 edges (which is 22.8% of the total possible edges.) The average degree of the graph, using all different possible perturbations 14.7. Restricted perturbation sets reduce the number of edges while keeping the number of perturbations fixed, making the graph sparser.

4.2 Model

We use a bi-directional LSTM trained on character-level embeddings. Embeddings for characters use the 64 most frequent characters based on the training set of SST-2. A biLSTM uses a single layer with an activation, and contains two channels through which information passes. The first channel corresponds to the output of the previous layer (called the hidden state), while the second channel theoretically captures longer term dependencies [6]. Since the LSTM is bidirectional, information propagates in both directions, and the concatenation of the two hidden states is used for final classification.

5 Experiments

Our primary experiments are as follows. First, for each sentence, we construct a graph based on perturbations of a specific word. We then run each perturbation of the initial sentence through the model to see if it is able to change the prediction. This defines labels for our label propagation algorithm.

We experiment with graphs over the same set of vertices (all edit distance one perturbations of the chosen word,) but vary the type of edges the graph can have. We also vary the percentage of data that receives initial labels (changes prediction vs does not change prediction) in the label propagation algorithm, which helps approximate the applicability of our approach in various settings.

All code used to produce the following results can be found here[1].

5.1 Primary Results

A summary of our results can be found in Table 1. In general, even when the percentage of labeled vertices is small, the accuracy of the label propagation algorithm is relatively high. This indicates that there is some signal, from the graph we construct, that can be used to discern whether or not an example will be adversarial or not. This can potentially save a significant amount of computational power when attempting to find large numbers of adversarial examples at once.

[1] https://github.com/ejones313/CS224W-GraphAdvExamples

Sentence	Perturbed Word	Edge Type	20% Labeled	50% Labeled
"this movie was fine"	*"fine"*	Insertions	52.3%	81.5%
		Deletions	47.7%	82.3%
		Substitutions	73.8%	83.8%
		All	77.7%	84.6%
"a fantastic movie"	*"fantastic"*	Insertions	60.0%	78.7%
		Deletions	52.6%	78.2%
		Substitutions	77.2%	85.9%
		All	80.0%	87.7%
"a movie with unfocused acting"	*"unfocused"*	Insertions	59.5%	86.2%
		Deletions	57.7%	82.1%
		Substitutions	80.8%	90.0%
		All	81.3%	91.8%
"what a cumbersome plot"	*"cumbersome"*	Insertions	68.6%	84.2%
		Deletions	63.1%	85.7%
		Substitutions	76.0%	86.9%
		All	83.5%	88.5%

Table 1: Table containing the primary results of this project. For each sentence we choose a word to perturb, and construct a graph with edges based on a specific type of perturbation. We report results when 20% and 50% of perturbations initially receive labels.

Sentence	Perturbed Word	L.C. Perts
"this movie was fine"	*"fine"*	39.3%
"a fantastic movie"	*"fantastic"*	33.3%
"a movie with unfocused acting"	*"unfocused"*	23.5%
"what a cumbersome plot"	*"cumbersome"*	33.5%

Table 2: Table showing the percentage of perturbations that are label changing (L.C.) for each sentence. *"This movie was fine"* has the greatest number of perturbations, while *"A movie with unfocused acting"* has the fewest. It's harder to achieve high accuracy when the percentage of label changing perturbations is higher.

Between different graphs, we find that graphs with edges between all perturbations perform best, which is unsurprising since they contain the most structural information. Interestingly, however, substitution-based graphs perform nearly as well as graphs containing edges corresponding to all edit distance one perturbations. Both of these edge frameworks outperform insertion and deletion based edges, especially when 20% of the data is labeled, rather than 50%.

To expand our analysis further, in Figure 1, we plot the accuracy of the label propagation algorithm as a function of the percentage of data that starts labeled. For each sentence, substitutions and all perturbations perform nearly identically, while deletions and insertions are noiser and perform worse. This indicates that substitution based neighbors of a token are more predictive of whether or not that token will be an adversarial example.

Performance of random classifiers. To interpret the accuracy numbers, it is important to consider the accuracy a random system would achieve. We can compute this using the percentage of perturbations that change the label for each sentence, which we provide in Table 2. The empirical between substitutions, insertions, and deletions that we discuss in the proceeding paragraphs appears more clearly in sentences with fewer perturbations that change the label, which are more representative of a general input. Though the sample size is low, this emphasizes that graphs with edges between substitution-linked tokens are useful in predicting whether or not a perturbation will alter a model prediction, which is a useful insight for future work in adversarial robustness in NLP.

6 Discussion

To conclude this paper, we will discuss potential applications of graphs in the adversarial robustness literature based on this project, and outline interesting directions for future work.

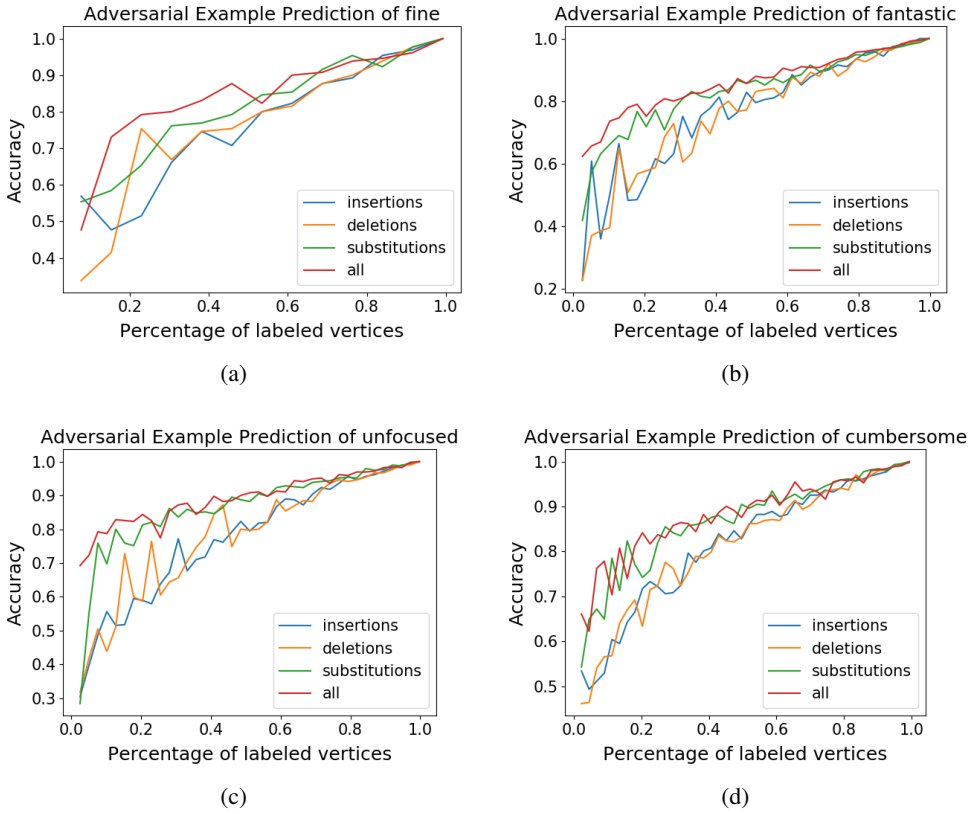

Figure 1: For each of the four sentences we consider, we plot the accuracy of the label propagation algorithm as function of the percentage of the total vertices that are labeled. Each plot contains all four types of edges. While all graphs converge (trivially) to perfect accuracy, all perturbations and substitutions outperform insertions and deletions, especially when the percentage of data that is labeled is small.

6.1 Applications

While several methods exist to find adversarial examples [4, 5], there has been little work considering the structure of the set of adversarial examples and in particular the relationship between them. In this work, we provide evidence that the space of adversarial examples using typos has some graph structure, and label propgation algorithms perform better on substitution-based graphs, as opposed to insertion and deletion-based graphs. In this sense, effectively projecting the space of all edit distance one typos to only the space of substitutions does not significantly reduce important structural information of the space, which is an interesting and somewhat surprising observation.

Application to certified robustness. Some work aimst to train models that provided *guaranteed* robustness, or robustness to every perturbation in some well-defined set. In particular, Pruthi et al [12] provides this guaranteed robustness by checking each perturbation in that set individually, which requires significant computational power.

However, a relaxed version of certified robustness requires checking some fraction of the perturbations before using our graphs to decide whether or not its necessary to check the rest. Though there are not explicit guarantees on the output of our algorithm, but such an approach could outperform existing heuristic approaches for searching through the perturbation space.

6.2 Future work

This work, due to time and resource constraints, leavs many directions unexplored. We enumerate a subset of these directions in the following paragraphs.

Application to different attack surfaces. In this work, we explore attacks to NLP models using adversarial typos. However, this is far from the only possible attack model. Another option is adversarial word substitutions: words are replaced with synonyms in an attempt to change the model prediction [8]. However, this surfaces lacks the substructure that typos have. A more promising application would be attacks in computer vision; if a model fails on a specific example, is it also likely to fail on an example less than ϵ away?

Testing over datasets. Currently, due to computational constraints, we only consider perturbations of a few sentences. Part of the reason such an approach is necessary is that checking all sentence level perturbations is intractable, since the number of those is exponentially large. Instead, we'd like to pick a single word, and consider all perturbations of that. One interesting way to extend our work to entire datasets is to use an interpretability method to determine which words in a sentence have the largest impacts on the final model prediction. We can then choose these words, and apply our analysis over a dataset, potentially strengthening the evidence of structure in the typo-space that we find in this work.

Moreover, label propagation is a nonconvex problem, so the choice of vertices to initialize may impact performace. We could thus improve our results by running the label propagation algorithm multiple times, then bootstraping to compute accuracy numbers.

Concluding thoughts. Though our work is limited by the relatively small sample size, we find evidence that graph structure is a useful feature in determining whether or not a perturbation will successfully fool a model. We hope that our approach is a step towards better understanding adversarial examples, why they appear, and what their structure looks like.

References

[1] Anonymous. Task-agnostic robust encodings for combating adversarial typos. In *Submitted to International Conference on Learning Representations*, 2020. URL `https://openreview.net/forum?id=rkgOuJBtPS`. under review.

[2] Shumeet Baluja, Rohan Seth, D Sivakumar, Yushi Jing, Jay Yagnik, Shankar Kumar, Deepak Ravichandran, and Mohamed Aly. Video suggestion and discovery for youtube: taking random walks through the view graph. In *International Conference on World Wide Web*, pages 895–904. ACM, 2008.

[3] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2018.

[4] J. Ebrahimi, A. Rao, D. Lowd, and D. Dou. Hotflip: White-box adversarial examples for text classification. *arXiv preprint arXiv:1712.06751*, 2017.

[5] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples, 2014.

[6] 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`.

[7] Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, Logan Engstrom, Brandon Tran, and Aleksander Madry. Adversarial examples are not bugs, they are features, 2019.

[8] Robin Jia, Aditi Raghunathan, Kerem Göksel, and Percy Liang. Certified robustness to adversarial word substitutions, 2019.

[9] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *CoRR*, abs/1609.02907, 2016. URL `http://arxiv.org/abs/1609.02907`.

[10] Yanpei Liu, Xinyun Chen, Chang Liu, and Dawn Song. Delving into transferable adversarial examples and black-box attacks. *CoRR*, abs/1611.02770, 2016. URL http://arxiv.org/abs/1611.02770.

[11] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. *CoRR*, abs/1403.6652, 2014. URL http://arxiv.org/abs/1403.6652.

[12] D. Pruthi, B. Dhingra, and Z. C. Lipton. Combating adversarial misspellings with robust word recognition. In *Association for Computational Linguistics (ACL)*, 2019.

[13] Aditi Raghunathan, Jacob Steinhardt, and Percy Liang. Certified defenses against adversarial examples. *CoRR*, abs/1801.09344, 2018. URL http://arxiv.org/abs/1801.09344.

[14] Mike Schuster and Kuldip K. Paliwal. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45:2673–2681, November 1997.

[15] R. Socher, A. Perelygin, J. Y. Wu, J. Chuang, C. D. Manning, A. Y. Ng, and C. Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Empirical Methods in Natural Language Processing (EMNLP)*, 2013.

[16] Florian Tramèr, Nicolas Papernot, Ian Goodfellow, Dan Boneh, and Patrick McDaniel. The space of transferable adversarial examples. *arXiv*, 2017. URL https://arxiv.org/abs/1704.03453.

[17] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. GLUE: A multi-task benchmark and analysis platform for natural language understanding. *CoRR*, abs/1804.07461, 2018. URL http://arxiv.org/abs/1804.07461.