Extracting Phone Numbers from Adversarial & Visually Corrupted Text

Timothy Forman and **Nathanael Chambers**
Department of Computer Science
United States Naval Academy
tforman37@gmail.com, nchamber@usna.edu

Abstract

Adversarial text is written with obfuscated words and characters that fools machine learned extractors in what would otherwise have been normal language for extraction. Social media often employs obfuscation for entertainment and style. Illicit domains like human trafficking use it in online advertisements, and recent work proposed the use of neural CRFs with character image recognition to extract phone numbers from such noisy text. However, that work simplified the task to reading short text snippets, rather than full pages. This paper proposes a new method to use prior adversarial work within a full real-world document task, and shows that small algorithm extensions can achieve similar performance. Further, we experiment with state-of-the-art visual recognition models and unicode injection techniques. Results show steady performance despite increased task difficulty, and only a 6% drop in accuracy when extracting from full documents rather than short snippets.

1 Introduction

Obscured text is often written to fool automatic extractors, but it also appears in social media simply as an expression of personal style (replacing the letter 'O' with an emoticon, or 'leetspeak' of the 1980's). While the former's usage is more deceptive in its motives, both are 'adversarial' in nature to an automatic extractor. This paper is relevant to both uses, but we focus on the illicit domain of human sex trafficking in its evaluations (and as an important societal use case). While our evaluation domain is specific, the broader task of extracting from *adversarial* noisy text is a general challenge for the NLP community.

Most recently, Chambers et al. (2019) proposed a *visual* neural conditional random field (CRF) that identified phone numbers in adversarial text. An example of an obfuscated text snippet from their challenge data is shown here:

3wõn7_28tree(øne)_573

The obfuscations in this text snippet include character visual swaps, homonyms, and noisy separators. This prior work showed how to use a *visual* character model, training a CNN to recognize each character by its 34x34 image, and then generalizing to recognize unicode character confounders. However, their models and evaluations simplified the task by only looking at bounded snippets where the system assumes the entire snippet holds a single phone number. This paper instead shows how to extend an extractor to apply first a longer padded snippet, and then second, a full document. Our new task is thus more complete and realistic; below is the above example in the padded setting (including a '140' distractor):

no strings 140 a friend 3wõn7_28tree (øne)_573 call me 62yo safe 4 you

Since the phone number does not begin at the start, training a model requires a variety of diverse training examples. Further, a simple model does not transfer to longer advertisements where most of the text is *not* a phone number. We first show that this more difficult padded setting can be solved with similar results to the easier bounded task simply by training on padded text. Second, we show how to transfer such a noise-aware model to full document text extraction with a sliding window detection approach, then followed by the standard phone number extractor. Despite moving from phrases to full documents, extraction performance only loses 6% in full phone number accuracy.

The main contributions in this paper are (1) the first adversarial phone number extraction from full documents, (2) results on a sliding window algorithm that show minimal loss, (3) experiments on integrating a SOTA hand-writing recognizer, and (4) a new dataset of full advertisements rather than short snippets.

Snippets (Chambers et al., 2019)	**289nu&VeI68500**
Padded Snippets	ogle Daves Group for all I **289nu&\eI68500** stream now - m4t (sling to
Full Advert.	Hi I'm rain I'm mobile looking for serious upscale men that want some companion ship ... I'm 5 8 and 140 thick 40ddd pretty feet long hair hmu u won't be disappointed **289nu&VeI68500** I'm only no Detroit area s I'm young tight ready to play

Table 1: The 3 types of datasets for this task: second and third are new to this paper and the focus of experiments.

2 Previous Work

This paper builds on the phone number extraction work in Chambers et al. (2019). They proposed a neural CRF over characters, and showed how to integrate 34x34 images of their character set into a CNN image input to their model. As the first to do this task, they limited it to short snippet extraction where the bounds of the snippet exactly includes the phone number. They concluded with a pilot experiment on full ad text and a rudimentary "padded" model. This paper extends their pilot results, evaluating a range of padded models on a new full-text dataset. An earlier system, TJBatchExtractor, used a rule-based regular expression setup (Dubrawski et al., 2015). It has been used for trafficking ID (Nagpal et al., 2017) and other works primarily with *non-obfuscated* text.

More broadly, the sex trafficking domain has focused more on linking advertisements through mentioned entities (Szekely et al., 2015) or training classifiers for types of trafficking ads (Alvari et al., 2016, 2017). Phone numbers are core inputs to most systems, but they assume extraction is given, and many found phone numbers to be important features (Dubrawski et al., 2015; Nagpal et al., 2017; Li et al., 2018) or even treated them as gold answers for prediction (Rabbany et al., 2018; Li et al., 2018). Phone numbers are one of the most stable links to entities (Costin et al., 2013).

Finally, a lot of work in the image community is relevant, and our model uses an image database of 65k unicode characters developed by BBVA Next Security Lab (github.com/next-security-lab) for phishing prevention. Related work uses CNNs for Asia-language classification (Liu et al., 2017; He et al., 2018). We also use data augmentation (Ding et al., 2016; Xu et al., 2016) to train the visual models. This is commonly used to learn robust recognizers (Salamon and Bello, 2017; Zhong et al., 2017).

3 Characteristics of Obfuscation

We briefly summarize the types of obfuscation that are seen in adversarial text, but refer the reader to Chambers et al. (2019) for a complete summary. Most obfuscations fall into 6 categories:

	Digit(s)	Obscured Text
Digits as Lexemes	4	FOUR
Homophones	24	twenny fo
Letters as Digits	1	I
Visual	8	8
Separators	410	4–1 _ 0
Reasoning	4	add(3+1)

The majority are substitutions of normal digits (4) into long form words (four or fore), visual lookalikes (unicode version of 8), or the insertion of distractors between digits. The visual deception is most difficult because they may be out-of-vocabulary or rarely seen in related contexts. Prior work proposed a CNN visual model to recognize character shapes, and this paper continues with those architectures.

While any one of these challenges might be 'solved' independently, the combination of these in a single text makes the task extremely difficult. Further, the main prior work on this topic used a simplified version where the input is just the obscured phone number, rather than a longer piece of text with the phone number hidden inside. This paper thus includes another type of obfuscation: an unknown location for the number. This paper is novel by both finding the number *and* extracting it.

4 Datasets

Table 1 shows an example of each dataset. The first is *Snippets*, taken from Chambers et al. (2019). This is the base task where the snippet is just the phone number. We use the same training set of 100k valid phone numbers that were randomly permuted with a variety of obfuscation operations. Because real-world examples are too few for neural training, this is an artificially generated training set. Most are more difficult to read than real-world examples, hence it allows the model to generalize to unseen real text.

We developed a second dataset called *Padded Snippets*, modifying the above *Snippets* by expanding each snippet to character length 70 by adding real-world text to its left and right with space separators. The text was randomly pulled from real

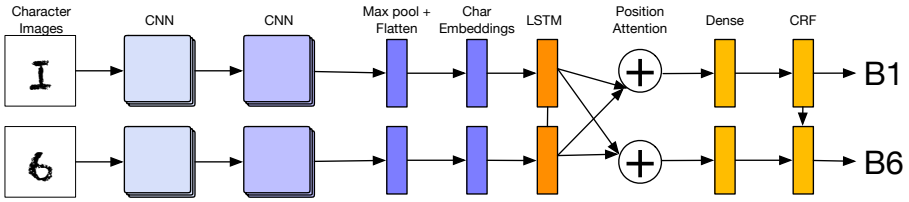

Figure 1: Abbreviated diagram of a 70-char network (only 2 of 70 inputs shown) to an LSTM with CRF predictions. Here the 'I' is predicted to be B1, beginning a 1 digit, and '6' to B6. Models without image input begin at the **Char Embeddings** layer.

advertisements to ensure realistic noise on both sides. The amount on the sides was also randomized, ensuring the number's location to vary across examples. The result is 100k padded snippets.

Finally, the third dataset unique to this paper is the *Full Advert* Dataset, comprised of *real* ads scraped from Backpage and Craigslist, and gold phone numbers identified by volunteer annotators. Again, see Table 1 for examples. All data is available for download [1].

5 Models for Snippet Extraction

Our baseline is the best performing model from Chambers et al. (2019). They showed that traditional NLP models with trained character embeddings struggle on unseen characters, so the input to this baseline is 34x34 *images* of characters. A stack of CNN layers transform each into a character embedding of size E, which is then input to a bi-directional LSTM whose hidden states are fed into a top-level CRF. CRF predictions use the standard BIO format on 10 possible digits with an Other category. Figure 1 shows a partial visualization of this model.

Using the above, we then train on *padded* inputs (Section 4) to see if the CRF can simply learn the added complexity. Further, we propose an improvement to its image recognition: instead of training our CNNs, use the state-of-the-art from the image recognition community. EMNIST is a handwriting task to recognize letters and digits. We used a recent EMNIST system as a black-box (github.com/shubhammor0403/EMNIST): the input being our characters, with the output a 62-dimension vector of probabilities (52 letters, 10 digits). This vector becomes the embedding for the character instead of training our own CNNs (inserted at 'Char Embeddings' of Figure 1).

We trained three variants of this EMNIST input: the first uses the 62-d vector as untrainable input embeddings, and the second adds a trainable dense layer of size E on top of the 62-d input. The dense layer(s) gives the model the ability to fine-tune dense weights over the non-trainable 62-d input. We tested a 2-layer deep version of this, and a third version that concatenates it to our own learned CNNs.

Document Extraction

One of the difficulties for *full* document extraction is the lack of training data, something that plagued the snippet extractors above. We don't have enough examples to train a full-document CRF, and applying a snippet-trained model outputs too many false positives. Advertisements are much longer, so the model needs to learn to ignore most inputs, as well as numeric distractors that appear, such as age, height, and weight. Rather than generating another artificial dataset, we instead propose a window span algorithm that can plugin any blackbox phone number extractor.

Our approach to window span identification splits the document into windows of text, and we then ask the extractor to output a phone number from each window. The extractor includes its overall probability, so we identify the window with highest probability of containing a phone number. This is our window identification step. We compute $P(phone|span) = \prod_{i=0}^{9} max_j P(d_i = j|span)$

A document's span with the highest probability is our target span. Once identified, the remaining task is simply phone number extraction as applied by the previous section's models.

6 Experiments

All extraction models were trained on our new 100k artificial *Padded Phone Snippets* (Section 4). 90k were used for training and 10k to determine convergence. The set of unicode characters used for training and testing were distinct from one another, to determine if the models were truly generalizing on the character's visual features. Based on development set optimization, all models use character embedding size *E=200* and LSTM internal

[1] www.usna.edu/Users/cs/nchamber/data/phone/

	no unicode		+ 30% unicode	
	Leven	**Perfect**	**Leven**	**Perfect**
RNN+CNN	77.2	42.9	70.3	19.4
CRF	**79.2**	49.8	55.5	0.7
CRF+CNN	79.0	**53.3**	**73.5**	**27.4**
CRF+EMNIST	79.0	52.9	54.3	2.0

Table 2: Padded Snippets results with unseen unicode injections.

	DEVELOPMENT				TEST			
	no unicode	10%	30%	50%	no unicode	10%	30%	50%
CRF	50.0	23.9	0.0	0.0	38.3	12.2	1.0	0.0
CRF+CNN	**52.4**	**43.7**	**27.9**	**19.9**	**39.3**	**31.8**	**21.4**	**10.5**

Table 3: Window ID + Extraction combined results reported as perfect accuracy %

dimension size *D=200* with a training dropout of 0.25. CNN convolution sizes of 4 and 8 were used respectively for the two CNN layers.

We report results with two metrics from prior work: Levenshtein edit distance and perfect accuracy. Levenshtein distance counts edits to turn the predicted phone number into the correct answer. This is more appropriate than digit accuracy because the CRF can output more or less than 10 digits. Perfect accuracy is how many phone numbers were predicted correct in all 10 digits.

Real-world Test: We report results comparing Snippet performance on the development set, but then report full document performance only on the *real-world* test set. We did not run models on the test set until the very end after choosing our best settings on the development set. These are unseen adversarial numbers.

Real-world Unicode Test: We also inject unseen unicode into the test set. This further tests a model's generalization. Using a hand-created character lookup of visually similar unicode characters, we replace varying $X\%$ of the adversarial text with unicode lookalikes not seen in training.

Finally, all reported results are the average of 4 trained/tested models of the same architecture.

7 Results and Discussion

The CRF with CNN performed best on our new Padded Snippets dataset, see Table 2. Our new results mirror relative performance from prior work, but now on the more difficult padded snippets, suggesting that our training with pads was sufficient. Integrating EMNIST failed to capture characters (2% vs 27.4% CNN), suggesting that EMNIST handwriting is too dissimilar from our unicode recognition challenge. Most important, Chambers et al. (2019) reported 58.1% perfect on 'easy snippets', only 6% higher than our 53.3% on the more difficult Padded Snippet task.

The full document results (see Table 3) are the first real-world advertisement results for adversarial phone number extraction. We ran the best CRF and CRF+CNN models from the development set results on this full document task. Even when requiring window search before extraction, extraction on ads with no injected unicode is 52.4% compared to padded snippets at 53.3%, showing a very small drop in performance due to window identification. On the test set, CRF+CNN extracts 31.8% of phone numbers from a 10% unicode document, compared to a meager 12.2% without CNNs. Window ID results were 95-99% accurate in finding the text span, so extraction performance changed by only 1-4% from Padded Snippet to Full Advert depending on the model used.

Previous work on adversarial phone number extraction showed good performance on a constrained task: bounded snippets. We have shown that constrained models can be adapted to full real-world document extraction by expanding training to padded snippets, utilizing injected unicode in training, and using a window identification algorithm prior to extraction. We will release all code for future replication, as well as the new datasets.

Conclusion

Social media is full of adversarial text, both incidental and intentional. While the societal impact of our particular domain, human sex trafficking, is critically important, our results are generalizable to other online communication domains as well. Unicode and emoticon usage will continue to expand, so visual models of language will grow in importance. We hope our results inspire future work on adversarial text extraction.

8 Acknowledgments

This work would not be possible without the help of the Global Emancipation Network, nor the financial and physical support of the DoD HPC Modernization Office by enhancing our undergraduate education and research. Finally, thanks to the Maui HPC Center for their support of midshipmen research.

References

Hamidreza Alvari, Paulo Shakarian, and J. E. Kelly Snyder. 2017. Semi-supervised learning for detecting human trafficking. In *Semi-supervised learning for detecting human trafficking*.

Hamidreza Alvari, Paulo Shakarian, and J.E. Kelly Snyder. 2016. A non-parametric learning approach to identify online human trafficking. In *IEEE Conference on Intelligence and Security Informatics (ISI)*.

Nathanael Chambers, Timothy Forman, Catherine Griswold, Kevin Lu, Yogaish Khastgir, and Stephen Steckler. 2019. Character-based models for adversarial phone extraction: Preventing human sex trafficking. In *Proceedings of the 5th Workshop on Noisy User-generated Text (W-NUT 2019)*, pages 48–56.

Andrei Costin, Jelena Isacenkova, Marco Balduzzi, Aurélien Francillon, and Davide Balzarotti. 2013. The role of phone numbers in understanding cyber-crime schemes. In *2013 Eleventh Annual Conference on Privacy, Security and Trust*, pages 213–220. IEEE.

Jun Ding, Bo Chen, Hongwei Liu, and Mengyuan Huang. 2016. Convolutional neural network with data augmentation for sar target recognition. *IEEE Geoscience and remote sensing letters*, 13(3):364–368.

Artur Dubrawski, Kyle Miller, Matthew Barnes, Benedikt Boecking, and Emily Kennedy. 2015. Leveraging publicly available data to discern patterns of human-trafficking activity. *Journal of Human Trafficking*, 1.

Linchao He, Dejun Zhang, Long Tian, Few Han, Mengting Luo, Yilin Chen, and Yiqi Wu. 2018. Visual-based character embedding via principal component analysis. In *International Conference of Pioneering Computer Scientists, Engineers and Educators*, pages 212–224.

Lin Li, Olga Simek, Angela Lai, Matthew P. Daggett, Charlie K. Dagli, and Cara Jones. 2018. Detection and characterization of human trafficking networks using unsupervised scalable text template matching. In *IEEE International Conference on Big Data (Big Data)*.

Frederick Liu, Han Lu, Chieh Lo, and Graham Neubig. 2017. Learning character-level compositionality with visual features. In *ACL*.

C. Nagpal, K. Miller, B. Boecking, and A. Dubrawski. 2017. An entity resolution approach to isolate instances of human trafficking online.

Reihaneh Rabbany, David Bayani, and Artur Dubrawski. 2018. Active search of connections for case building and combating human trafficking. In *KDD*.

Justin Salamon and Juan Pablo Bello. 2017. Deep convolutional neural networks and data augmentation for environmental sound classification. *IEEE Signal Processing Letters*, 24(3):279–283.

Pedro Szekely, Craig Knoblock, Jason Slepickz, Andrew Philpot, et al. 2015. Building and using a knowledge graph to combat human trafficking. In *International Conference on Semantic Web (ICSW)*.

Yan Xu, Ran Jia, Lili Mou, Ge Li, Yunchuan Chen, Yangyang Lu, and Zhi Jin. 2016. Improved relation classification by deep recurrent neural networks with data augmentation. *arXiv preprint arXiv:1601.03651*.

Zhun Zhong, Liang Zheng, Guoliang Kang, Shaozi Li, and Yi Yang. 2017. Random erasing data augmentation. *arXiv preprint arXiv:1708.04896*.