CS229

Identifying Human Trafficking: Using Machine Learning to Detect Trafficking from Online Sex Ads

Josh DuFault
Department of Computer Science
Stanford University
jdufault@stanford.edu

Abstract

Human sex traffickers will post online sex ads that are purportedly from willing sex workers, but are actually advertising the services of a trafficking victim. In this project, I use a CNN over the ad text to create a trafficking/not-trafficking predictive model and use logistic regression over a set of extracted ad features to identify the most predictive features.

1 Introduction

This Summer I worked for Stanford's Hazy Research lab, writing feature extractors for a webcrawl of sex ads. The extracted features were part of a larger project to create a data visualization tool to help law enforcement combat sex trafficking. For a CS221 project, I continued work on feature and relation extractions to develop a structured knowledge base from the unstructured ad text. Feature extraction and dataset preparation was either done for CS221 or done previously. Bucketing and vectorizing features, as well as all modeling, were done for CS229.

This project tests the performance of machine learning models using the raw ad text as input and traditional machine learning models including logistic regression and SVMs that use the extracted features as input. It then uses the most effective combination of models to create a trafficking/not-trafficking prediction model that can be run on the entire ad base. This will be used as input to the data visualization tool. It will also be used to estimate the overall prevalence of trafficking in online sex ads. The weights of the extracted features will be used to identify the most predictive features.

2 Related work

In *A Non-Parametric Learning Approach to Identify Online Human Trafficking*, Alvari et al. use human domain experts to identify sex-trafficking instances by reading ads. They then use this gold dataset to predict trafficking [1]. In this project, I am using a gold dataset created using identifiers found during law enforcement investigations. I could not find similar work with this type of high-precision dataset on Google Scholar.

It is difficult to create a word based vocabulary for these online ads because they make frequent use of abbreviations, use coded terms, and have misspellings. For example, the term "qv" means quick visit. For this project, I explore using a character based encoding. There are several papers exploring language modeling and binary classification at the character level. The type of network that gave the highest performance was a combined network with a CNN layer that fed into a LSTM layer [2].

CS229: Machine Learning, Fall 2018, Stanford University, CA. (LateX template from the CS230 website.)

There has also been work exploring the use of deep CNNs at binary classification of Twitter tweets which, like online ads, are similarly length constrained [3].

3 Dataset

The ads come from a webcrawl of over 1 billion sex ads. These are saved as raw HTML and the ad text is extracted from them. Due to infrastructure limitations, a subset of approximately 20 million ads was used. The positive and negative datasets were created from this subset. See Figure 1 for an example of the extracted ad text.

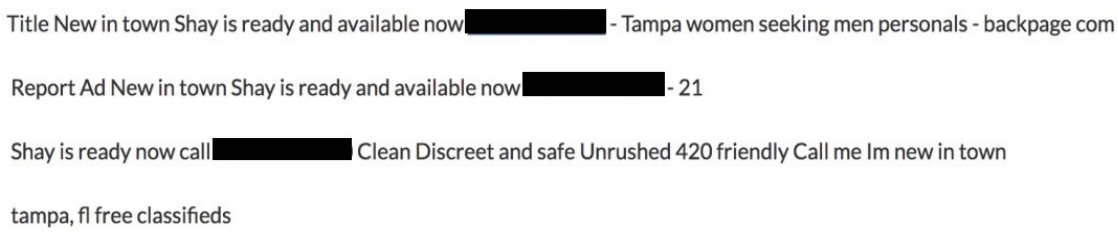

Figure 1: Extracted text of ad.

The project also has a list of phone numbers that have been positively associated with sex trafficking during law enforcement investigations. 5,586 of these phone numbers were present in the subset of ads. These 5,586 ads form the positive examples in the gold dataset. Deduping using locality sensitive hashing to detect ads with Jaccard similarity ≥ 0.9, left 2,840 unique ads.

6,000 different ads were chosen at random from the same subset. These ads form the negative examples in the gold dataset. Deduping using locality sensitive hashing to detect ads with Jaccard similarity ≥ 0.9, left 5,163 unique ads.

4 Features

4.1 Linear classifier features

The extracted features consist of: ad text, city population, city latitude, city longitude, price per hour, price per half hour, contact information, age, ethnicity of the poster, date/time/day of week of ad posting, and in-call or out-call services. I also used indicators for whether an ad contained: specific mentions of any price, mentions of an in-call establishment, specific mentions of sexual services, the number of ads with Jaccard similarity of 0.9 or greater as a proxy for the number of duplicate ads, and the number of ads with Jaccard similarity of 0.5 or greater as a proxy for the number of different ads from the same author.

Features were bucketed before being used as input to the logistic regression classifier. City population was specified in increments of 10,000. Geocodes were rounded to the nearest degree, bucketing them into squares of approximately 70 miles. Only the area codes were used from extracted phone numbers. Time of posting was split into day, night, and graveyard shifts.

4.2 Neural network features

The extracted text of the ads without any extracted features was used as input. Each character was used to create a one hot encoding where the vector size (90) was the size of every type of character present in the ads plus a special character to represent a padding character: '>'. The ad text was truncated at 2,000 characters by the text extractor and padded to a length of 2000 characters for shorter ads. The character vectors were then converted into embedded vectors of size four. The embedding size was chosen by taking the fourth root of the vocabulary size and rounding up. This was based on the recommendation from Google's TensorFlow team [4].

5 Methods

5.1 Logistic regression

Logistic regression was run over the extracted features. Logistic regression uses the logistic function, $h_\theta(x) = \frac{1}{1+e^{-\theta^T x}}$, to output a hypothesis from 0 to 1. This provides a convenient interpretation of the results as probabilities which is useful because the goal is to predict the probability that an ad is an example of human trafficking. Gradient ascent is then used to maximize the log likelihood: $L(\theta) = \sum_{i=1}^{m} y^{(i)} \log h(x^{(i)}) + (1 - y^{(i)}) \log(1 - h(x^{(i)}))$. Where $L(\theta)$ is derived by assuming $P(y = 1|x; \theta) = h_\theta(x)$, $P(y = 0|x; \theta) = 1 - h_\theta(x)$, and that all m training examples were generated independently [5].

5.2 SVM

A support vector machine was also run over the extracted features. Support vector machines are linear classifiers like logistic regression, but have the advantage of fitting a decision boundary that maximizes the geometric margin. This results in a classifier that maximizes the gap between the positive and negative examples. Additionally, SVMs can be used with kernels which replace $\langle x, z \rangle$ with $K(x, z)$ in the optimization problem. The kernel corresponds to a feature mapping θ, thus: $K(x, z) = \theta(x)^T \theta(z)$. In this case, the Gaussian kernel, $\exp(-\frac{\|x-z\|^2}{2\theta^2})$, was used providing an infinite dimensional feature mapping [6].

5.3 CNN

In a convolutional neural network, multiple filters of a specified size are trained over the entire model. This model used three CNN layers each followed by a 1D max-pooling layer of four. Each CNN layer used filters of size 16 to identify relevant features in the character vector and a RELU activation function. The filter size was chosen based on the typical size of a long word in the ads. This was to ensure that the characters could be easily mapped to words. The max-pooling layers reduced the dimensions of the large input character vectors while identifying only the most significant features in a convolved section. It was then followed by an additional max-pooling layer and a fully-connected layer with a sigmoid activation function. The loss function was binary cross entropy with Adam optimization and dropout regularization. See Figure 2 for a visualization of the model.

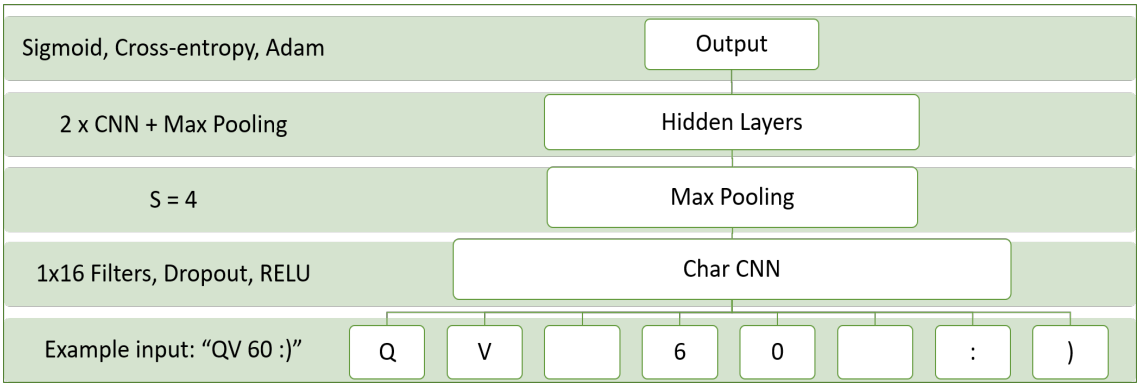

Figure 2: CNN diagram of input specifying a price.

5.4 CNN + LSTM

In a long short-term memory neural network, each node takes input from both its own character vector the previous node's vector. This model used one CNN + max-pooling layer with the same parameters of the deep CNN. The CNN layer was then followed by a LSTM layer. A tanh activation function was then applied to the output of each node. Each node in the LSTM was then fed into a

fully-connected layer with a sigmoid activation function. The diagram of this model is the same as the deep CNN's diagram except with the hidden layers replaced with a single LSTM layer.

6 Experiments/Results/Discussion

Due to the relatively small size of the dataset, I used a 70/30 train/test split. For the neural networks, I set a learning rate of 0.001 based on the suggested learning rate in the original paper proposing the Adam stochastic gradient descent optimization [7]. I initially set the mini-batch size to 128 as used in the Adam paper. However, I found that reducing the mini-batch size to 32 allowed the CNN to reach .91 accuracy in only two epochs, significantly reducing training time.

Initially, the logistic regression model had .94 accuracy. However, much of that predictive ability came from the weekday the ad was posted and the area code of the poster. This can be explained by bias in the data as the subset of 20 million ads were a time slice of the larger set of 1 billion ads and the positive dataset was made using data from cooperating agencies which have geographical jurisdictions. Therefore, these features were excluded.

The model using the LSTM had the worst performance. I believe this was because the long character vectors made it difficult for the gradients to propagate through the time-steps correctly. Further hyper-parameter adjustment could improve the performance of the CNN+LSTM model; however, due to the slow inference time and good performance of the CNN, I chose not to pursue the LSTM based models.

The primary performance metrics of interest are test accuracy and inference time. This is because the positive dataset is small relative to the entire dataset of 1 billion ads. The goal is to create a model that can be run on the entire 1 billion ads so training time will be far smaller than the overall inference time. Based on this, the CNN and logistic regression are the most promising. See table 1.

Model	Train accuracy	Test accuracy	Train time	Inference time
Logreg	.81	.79	<1s	**<1s**
SVM	.90	.84	3s	2s
CNN	.91	**.93**	43s	5s
CNN+LSTM	.66	.65	217s	23s

Table 1: Comparison of model performance.

The primary difference between the logistic regression model and the SVM was the decreased number of false positives in the SVM. See Table 2 and Table 3. The CNN outperformed both linear models in every category. See Table 4.

n=2401	Predicted negative	Predicted positive
Actual negative	1337	239
Actual positive	251	574

Table 2: Logistic regression confusion matrix.

n=2401	Predicted negative	Predicted positive
Actual negative	1463	113
Actual positive	279	546

Table 3: SVM confusion matrix.

n=2401	Predicted negative	Predicted positive
Actual negative	1485	45
Actual positive	147	724

Table 4: CNN confusion matrix.

The CNN had the best performance by a solid margin. A potential point of concern with using a CNN model is that it's difficult to tell which features the model is using to make its predictions. Initially, the logistic regression model had very high accuracy due to biases in the data but this was easy to detect. This might not be the case with a CNN. However, as the sophistication of the linear models increased, the accuracy approached that of the CNN which at the least makes the CNN's performance appear reasonable. See Figure 3.

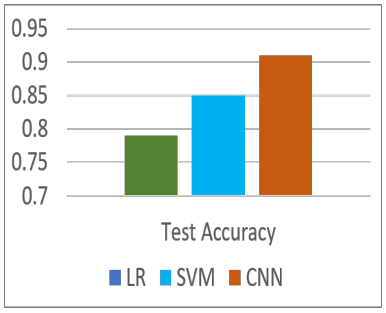

Figure 3: Comparison of linear models and CNN.

An inspection of the false positive predictions by the CNN, show that the ad text extraction is partly responsible. The majority of false positives came from poorly formatted extractions. For example, many websites include similar ads next to the main ad and the extractor sometimes concatenated them together. In other instances, it extracted only the title. An inspection of the false negatives revealed no obvious patterns.

The CNN does not appear to be overfitting to the training set because the train accuracy and test accuracy are similar. Dropout regularization was used to prevent overfitting. The logistic regression model overfits slightly.

The coefficients of the logistic regression model can be used to determine the most predictive features. In order, the top eight most predictive features are: the website the ad is posted on, if there is a price or rate listed, if there are sexual services explicitly listed, if there is an in-call location listed, the price per half-hour, the ethnicity of the poster, the number of similar ads (Jaccard similarity $\geq .5$), and whether the poster offers in-call or out-call services.

The most predictive features as indicated by logistic regression, create a profile of a particular type of ad most associated with human trafficking, e.g. "come to our spa, full nude massage, 60/hour," etc. However, this does not indicate that all such ads are associated with trafficking. In particular, the overall prevalence of trafficking was not investigated for this project and a balanced dataset was used to train the models.

7 Conclusion/Future work

I believe the performance of the models is high enough to justify further research. Next, I would like to create a larger dataset over a larger section of time to test the CNN on. Training the CNN on several years of data and then testing it on a different year to see if it is still able to make accurate predictions, would give confidence that the model is effective and not just detecting some unnoticed bias in the data. I would also like to run the model on a larger section of the ads to estimate the overall prevalence of sex trafficking in online ads.

8 Code

Project: https://github.com/joshduf/extractors/
Models referenced in this report are in the "prediction" folder.

References

[1] Alvari, Hamidreza, Paulo Shakarian, and JE Kelly Snyder. "A non-parametric learning approach to identify online human trafficking." *Intelligence and Security Informatics (ISI), 2016 IEEE Conference on.* IEEE, 2016.

[2] Jozefowicz, Rafal, et al. "Exploring the limits of language modeling." *arXiv preprint arXiv:1602.02410* (2016).

[3] Dos Santos, Cicero, and Maira Gatti. "Deep convolutional neural networks for sentiment analysis of short texts." *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers.* 2014.

[4] TensorFlow Team. "Google Developers Blog: Introducing TensorFlow Feature Columns." *Google*, 20 Nov. 2017, https://developers.googleblog.com/2017/11/introducing-tensorflow-feature-columns.html.

[5] Ng, Andrew. "Support Vector Machines." CS 229, 10 Oct. 2018, *Stanford University*. Lecture notes.

[6] Ng, Andrew. "Supervised Learning, Discriminative Algorithms." CS 229, 26 Sep. 2018, *Stanford University*. Lecture notes.

[7] Kingma, Diederik P., and Jimmy Ba. "Adam: A method for stochastic optimization." *arXiv preprint arXiv:1412.6980* (2014).

[8] Pedregosa, Fabian, et al. "Scikit-learn: Machine learning in Python." *Journal of machine learning research* 12.Oct (2011): 2825-2830.

[9] Chollet, François, et al. "Keras." (2015).