CS221

Identifying Human Trafficking: Using AI Techniques to Extract Predictive Features in 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 weak-supervision, Bayesian techniques, and locality sensitive hashing to identify and extract predictive features.

1 Introduction

Researchers have identified ad characteristics that are correlated with sex trafficking which can be used to create predictive models to classify ads as sex trafficking. Law enforcement can also use domain knowledge combined with the ad characteristics to identify sex trafficking. However, the ad text is unstructured, the authors may intentionally write the ads to avoid detection using key word searches [1], and applicable labeled datasets are rare or non-existent

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 CS229 project, I developed a trafficking/not-trafficking predictive model. The data visualization tool, the initial web-crawl, the positive dataset, and an initial set of feature extractors, were created by others on the project or by myself prior to CS221. The rest of this section describes work done by myself for CS221.

In this project, I use semi-supervised learning to extract the $price(\$\$\$, hour)$ relation, which is strongly correlated with sex-trafficking [2] and proved too variable to extract with a simple regular expression (regex). I also use Bayesian techniques to identify the key n-grams most strongly associated with sex-trafficking, and group those n-grams to extract additional features positively correlated with sex-trafficking. Additionally, I implement locality sensitive hashing to remove duplicates from the dataset and identify similar but unique ads in $O(n)$ time. Duplicate ads affected the performance of the trafficking/not-trafficking predictive model and similar ads are used to identify unique ads written by the same author. Finally, I adjusted the text extraction used to create the positive and negative datasets as that also affected the performance of the predictive models.

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 identify key phrases and features indicative of sex-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.

CS221: Artificial Intelligence, Fall 2018, Stanford University, CA. (LateX template from the CS230 website.)

In *Rational Pricing in Prostitution: Evidence from Online Sex Ads*, DeAngelo et al. describe how sex workers display economically rational pricing [2]. Unusually low prices are correlated with sex trafficking because trafficked victims don't have control of their own prices. Price extraction is necessary both to determine the median price for a location and identify outliers.

Human traffickers will deliberately write ads with in such a way as to defeat key-word searches by law enforcement [1]. Even non-trafficked sex-workers will attempt to obfuscate references to money to defeat keyword searches. This makes it difficult to extract the price relation using a regex. There is also no gold dataset to identify prices in online sex ads to create a machine learning model. In *Snorkel: Rapid training data creation with weak supervision*, Ratner and Ré describe a weak supervision framework, Snorkel, that can extract relations from unstructured text using a small labeled dataset.

Although this project is using datasets of $< 6,000$ ads due to limitations with the positive dataset, the entire webcrawl contains over one billion ads. To be able to run extractions at scale, they must run in linear time. A Bayesian learning algorithm reduces to a counting argument and can be done in linear time [4]. Further, locality sensitive hashing can be used to estimate the Jaccard similarity of documents in linear time [5]. In *Authorship Verification, Average Similarity Analysis*, Castro et al. identify Jaccard similarity using an n-gram based model as having similar performance in authorship identification as more complicated measures.

3 Dataset

The ads come from a webcrawl of over one 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.

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.

6,000 different ads were chosen at random from the same subset. These ads form the negative examples in the gold dataset.

4 Data cleaning

The ads are saved as raw HTML. Many hosting sites include other ads with similar characteristics in addition to the ad searched for. This made extracting the relevant text difficult. The project had two extractors to obtain the ad text: an HTML parser using Beautiful Soup, and a regex based extractor.

The Beautiful Soup extractor was too slow to run at a scale of millions. It also frequently extracted non-ad text, sometimes including pages of HTML commands. The regex extractor was both faster and higher precision but the recall was sensitive to the site the ad was downloaded from. The entire webcrawl consisted of 10 domains, so it was feasible to update the regex for all the domains. I ran 100 ads through the extractors and for each ad, awarded 1 quality point to the extractor with the best extraction. Ties received 0.5 points. I ran 1000 ads to estimate the time in seconds. Based on the results (Table 1), I decided to use the updated regex (Figure 1).

Extractor	Quality	Time (s)
Beautiful Soup	41	1180
Updated Regex	59	284

Table 1: Comparison of ad text extractors.

```
title_term = r'<[Tt]itle>([\w\W]*?)</[Tt]itle>'
body_term  = r'<div.{0,20}[Cc]ontent.{0,20}>([\w\W]*?)</div>'
body_term2 = r'<div.{0,20}[Pp]ost.{0,20}>([\w\W]*?)</div>'
body_term3 = r'<div.{0,20}[Tt]ext.{0,20}>([\w\W]*?)</div>'
body_term4 = r'<p>([\w\W]*?)</p>'
```

Figure 1: Updated regex

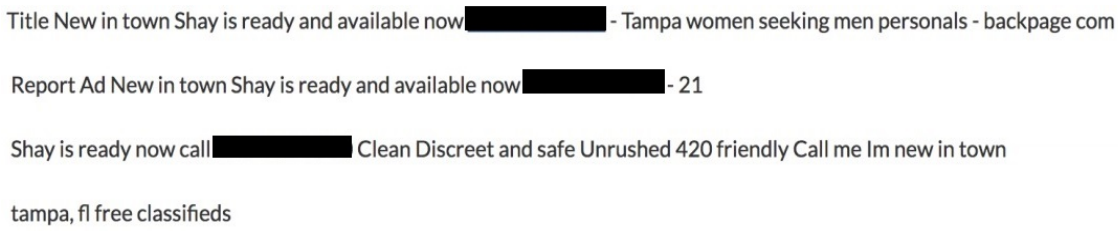

Figure 2: Extracted text of ad with indicators of trafficking.

5 Price relation extraction model

Here I extract the $price(\$\$\$, hour)$ relation.

The baseline is a regex extractor (Figure 3). This proved reasonably high precision but low recall and brittle. The authors of the ads would often use codewords to attempt to obfuscate the fact that money is involved. The baseline accurately identifies the first example in Figure 4 but misses the second example. It would be possible to update it to get the second example but since traffickers are specifically trying to write ads to defeat this type of naive search [2], a more robust method is needed.

```
num_reg = re.compile(r'^(\d?\d[05])$')
hour_reg = re.compile(r'\Whr\W|\Whour\W|60 min|\Wh\W')
half_reg = re.compile(r'half|hh|hlf|hhr|1\/2|30 min')
quick_reg = re.compile(r'qv')
```

Figure 3: Baseline price extractor regex.

```
Example 1: 100 roses qv 150 hh
Example 2: donations: 100, 30m: 150.
```

Figure 4: Text indicating a price of $150 per hour hour.

For this extractor, I use the Snorkel weak supervision framework developed in Stanford's Hazy Research lab (Figure 5). This framework allows a relation extraction model to be trained using only small amounts of labeled data and a handful of heuristics [3]. This is useful because there is no labeled data for this relation extractor.

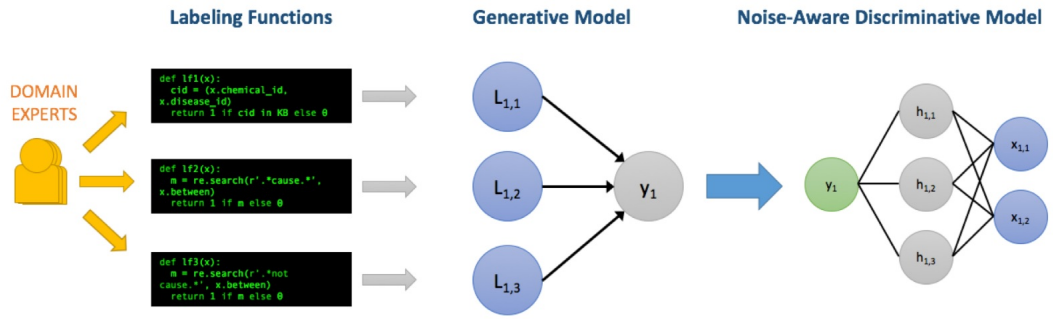

Figure 5: Snorkel workflow [4].

To apply Snorkel to this task I did the following:

1. Wrote a more permissive regex to identify any reasonable number that could indicate a price (Figure 6).
2. Ran Snorkel to identify candidates based on the regex.
3. Hand labeled 100 of the identified candidates as correct/incorrect.
4. Wrote heuristic labeling functions to identify the correct candidates (Figure 7).
5. Ran the Snorkel generative model to get weights for each labeling function.
6. Ran the Snorkel discriminative model to train an LSTM to classify each candidate.
7. Saved the candidates Snorkel identified as correct for each document.

```
(\W[1-5]?\d[05]\W)
```

Figure 6: Candidate extractor regex.

```
def lf_preceding_half(c):
    preceding_words = ['half', 'minutes']
    return -1 if overlap(
        preceding_words,
        get_left_tokens(c, window=1)) else 1
```

Figure 7: Example labeling function.

The Snorkel model has improved recall over the baseline using the sample of 100 hand labeled ads. It also has improved precision which is typically regex's strong suit. The regex extractor in particular does not handle non-standard grammar well. This is a problem because immigrants are targeted by sex trafficking groups and non-native English is a potential identifier. The oracle is a human expert which does unsurprisingly well as the ads are designed to be read by a person. See Table 2.

Extractor	Precision	Recall
Baseline	89	34
Snorkel	91	78
Oracle	100	100

Table 2: Comparison of $price(\$\$\$, hour)$ relation extractors.

6 N-gram identification

This model uses a Bayesian network with Laplace smoothing to identify the n-grams most strongly associated with human trafficking. In this scenario, I consider each n-gram as a Bayesian network where each word w_i is a variable with a factor conditioned on the words $w_1, ..., w_{i-1}$ that came before it:

$$p(w_1, ...w_n) = p(w_1)...p(w_{n-1})p(w_n|w_1, ..., w_{n-1})$$

For this project, n-grams of size $n \in \{1, 2, 3\}$ were used. N-grams larger than three became increasingly unlikely to appear in both datasets. This created a probability for all n-grams in the gold dataset of ads associated with human trafficking and a similar dictionary of probabilities in the negative ad examples. N-grams were then assigned a score proportional to the probability that each n-gram is associated with trafficking:

$$\frac{p(t)p_{td}(ngram)}{p(nt)p_{ntd}(ngram)}$$

Where t is trafficking, nt is non-trafficking, and d is dataset. The Laplace smoothing used above prevents dividing by 0.

The n-grams most strongly associated with sex trafficking were then separated into categories. See Figure 8. New feature extractors were created from the top three categories: whether an ad contained specific mentions of any price, mentions of an in-call establishment, and specific mentions of sexual services.

Figure 8: Top n-grams most associated with human trafficking separated into categories by color.

To evaluate the performance of these three extractors, I used the support vector machine (SVM) trafficking/not-trafficking prediction model developed for CS229. See Table 3.

SVM	Accuracy
Without n-gram based extractors	.79
With n-gram based extractors	.81

Table 3: Performance of SVM using n-gram based extractors.

7 Locality sensitive hashing

To identify the number of near duplicate and similar ads I did the following:

1. Turned the extracted text of each ad into a list of three word shingles. For example: "My name is Ashley, call me today," would become ["my name is", "name is ashley", "is ashley call", "ashley call me", "call me today"].

2. Ran each shingle through a set of ten hashes, keeping the minimum hash.

3. Returned a list of the ten minimum hashes in order as the document's fingerprint.

4. The number of hashes in the fingerprint that two documents share in common is approximately the Jaccard similarity of the two documents using n-grams of size three [5]. A dictionary of hashes to document IDs can then be checked and updated once per document, giving an approximately linear run time.

I then created extractors identifying whether each document is a duplicate and how many similar ads are in the dataset. To evaluate the performance of these extractors, I again used the support vector machine (SVM) trafficking/not-trafficking prediction model developed for CS229. See Table 4.

SVM	Accuracy
Without similarity metric extractors	.81
With similarity metric extractors	.84

Table 4: Performance of SVM using similarity metric extractors.

Unfortunately, there is no labeled dataset to detect if similar ads are actually by the same author. For an example of ads marked as duplicate or similar, see Figure 9, Figure 10, and Figure 11.

```
Working 11am to 4pm - Georgia therapeutic massage
Licensed massage therapist & Im also very gorgeous.
Give me a call. You will not be disappointed.
Have massage table & unscented oils Joy.
```

Figure 9: Original ad.

```
Working 11am to 4pm - Atlanta therapeutic massage
Licensed massage therapist & Im also very gorgeous.
Give me a call. You will not be disappointed.
Have massage table & unscented oils Joy.
```

Figure 10: Ad identified as near duplicate.

```
Working SATURDAY 11am to 3pm - Atlanta therapeutic massage
Licensed massage therapist & Im also very gorgeous.
Give me a call. You will not be disappointed.
Have massage table & unscented oils Joy.
```

Figure 11: Ad identified as similar.

The locality sensitive hashing similarity comparison is significantly faster than a naive implementation of Jaccard similarity that runs in quadratic time. In the figure below, the large dataset is twice the size of the regular dataset but takes four times as long to run. This quadratic scaling is clearly unsustainable for a billion ads. See Figure 12.

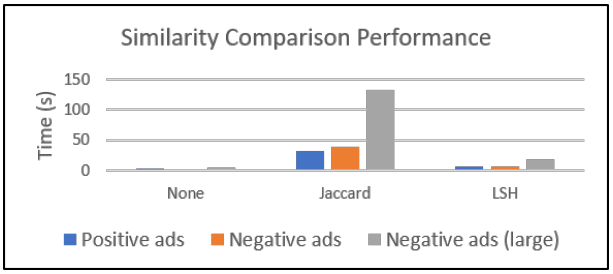

Figure 12: Similarity comparison times.

8 Conclusion/Future work

The final performance of the SVM (.84) leads me to believe that these features have a predictive value in identifying human trafficking. The n-gram identification identifies a particular profile of ad most associated with human trafficking, e.g. "come to our spa, full nude massage, 60/hour," etc. Snorkel performs well for relation extractions and is a strong improvement over regexes. The locality sensitive hashing comparison is fast enough to run at scale and appears to detect true and near duplicates well. Next, I would like to find a labeled dataset to get a better test of its accuracy at detecting authorship.

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] DeAngelo, Gregroy, et al. *Rational Pricing in Prostitution: Evidence from Online Sex Ads.* Working Paper, 2017.

[3] Ratner, Alexander, et al. "Snorkel: Rapid training data creation with weak supervision." *Proceedings of the VLDB Endowment* 11.3 (2017): 269-282.

[4] Liang, Percy. "Lecture 15: Bayesian Networks III." CS 229, 12 Nov. 2018, *Stanford University*. Lecture notes.

[5] Anand, Rajaraman, and D. U. Jeffrey. "Mining of massive datasets." 2011-01-03. *http://infolab. stanford, edu/ ullman/mmds/book, pdf* (2012).

[6] Castro, Daniel Castro, et al. "Authorship Verification, Average Similarity Analysis." *Proceedings of the International Conference Recent Advances in Natural Language Processing.* 2015.

[7] Ratner, Alexander, et al. "Data Programming + TensorFlow." *Snorkel Blog*, 15 Dec. 2016. Accessed 14 Nov. 2018.