Training AI to Detect Stolen Car Parts Online: Project Overview and Methodologies

Harmond Drenth, Kurt Wokoek, Warren Burrus, Courtney Hodge, Denny Lee, and Tacoma Velez

Principal Investigator: Dr. Pablo Rivas

Baylor University

GitHub Repo

RivasCars

Introduction

In the realm of online Consumer-to-Consumer (C2C) marketplaces, the illicit sale of stolen car parts is a significant issue. This project aims to use multimodal data (text, images, user data) to train machine learning models to detect and identify listings of stolen car parts, thus contributing to reducing online sales of stolen goods.

Methods

- Scraped thousands of Craigslist, OfferUp posts
- Used Python scripts to generate CSV files of post data
- Read research papers on illicit online sales
- Constructed risk score formula from research to evaluate suspicious posts
- Labeled dataset of 2,000 sample posts to train NER model
- Produced visualizations, geo-plots of location and city data on posts
- Answered client's research questions in Python Colab notebooks
- Detected red-flag words in posts using sentiment analysis and BLEU scores
- Documented all work, code, and findings in "RivasCars" GitHub repo

Results

- Answers to research questions
  - See Research Question and Data Visualizations.pdf

- Trends in post data (common red flags, prices, locations, etc.)
    - See Research and Risk Score Formula.pdf
- Risk Score Formula
    - See Research and Risk Score Formula.pdf
- Geo-plots and visuals
    - See Geo-Analytics folder
- Convergence of NER model
    - 60-70% performance of detecting labels in posts

Detecting Red Flags

We were unable to integrate red flag detection as part of our NLP model due to a lack of red flags in the dataset to label, and due to its impacting performance, we decided to scratch it as part of the NLP model. Instead, we investigated standalone solutions that were more customizable and modular to detect red flags in data. First, we will introduce to the reader what a red flag is.

As a refresher, a “red flag” is a colloquialism used to refer to a phrase in a post body that suggests that a part could be illicitly obtained, or to put it more simply, “fishy”. The challenge in detecting red flags is the wide variety of categories of red flags, the wide diversity in how they manifest themselves, and the context clues necessary to pick up on them. Some examples of red flags:

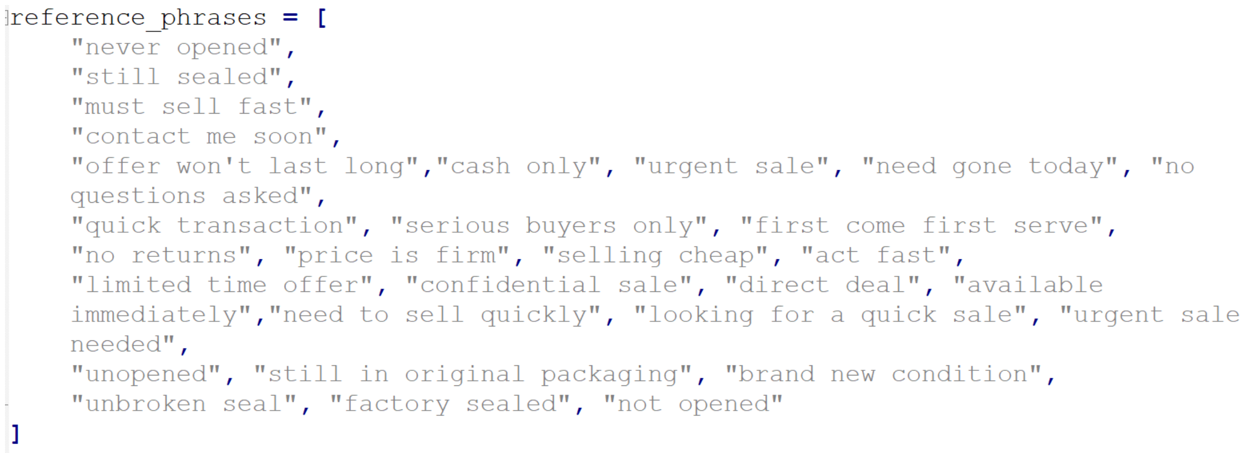
- “Brand new in packaging”
- “Must sell today”
- “No questions asked”
- “as is, no returns”

In the above, the aforementioned diversity of phrases can be seen. The other important fact to note is that context matters greatly with these. For example, if we saw that someone had a pallet full of engine parts for sale, still all in the wrapping, for a comparatively low price (say $10 when the items cost $100 each), one would be more suspicious than someone who had a vintage collector’s car part, sealed in the packaging, for a collector’s upcharge. What this means is that we cannot just automatically assume posts that contain “red flags” are indeed suspicious and worthy of further inspection, we need to consider them as a weighting in a more complex risk formula, as outlined in Research and Risk Score Formula.pdf. However, detection of red flags posed a more difficult problem than originally anticipated. It is not practical to use “dumb” matching based on keywords, like a regex check or keyword search, due to the vast amount of variation in wording and meaning. For this reason, our group decided to explore several different methodologies to detect these red flags in post text.

Firstly, we needed to establish a common framework to evaluate and compare methodologies. This consisted of a bank of handcrafted phrases deemed to contain red flags by the team, as well as several "red herring" phrases that contained no red flags, and even excerpts from literature. Next, we needed a way to feed the post body to the various methodologies to test. We decided on tokenizing each sentence of the post, using periods to delimit the body. That means that a post like this:
"Selling a like new hood for a 2001 Subaru Impreza. I can be contacted by phone at XXX-XXX-XXXX. Asking for $125 or best offer."

...would result in three tokens: "Selling a like new hood for a 2001 Subaru Impreza.", "I can be contacted by phone at XXX-XXX-XXXX", and "Asking for $125 or best offer."

Then, each tokenized sentence would be evaluated individually, with a score for each sentence being output. We decided on this bank of reference phrases, which represents a decent variety of concerning phrases:


```
reference_phrases = [
    "never opened",
    "still sealed",
    "must sell fast",
    "contact me soon",
    "offer won't last long","cash only", "urgent sale", "need gone today", "no
    questions asked",
    "quick transaction", "serious buyers only", "first come first serve",
    "no returns", "price is firm", "selling cheap", "act fast",
    "limited time offer", "confidential sale", "direct deal", "available
    immediately","need to sell quickly", "looking for a quick sale", "urgent sale
    needed",
    "unopened", "still in original packaging", "brand new condition",
    "unbroken seal", "factory sealed", "not opened"
]
```

The first methodology was using BLEU scores to determine the closeness of each sentence to our sentence bank. This gave good results, but did have a few downsides. In terms of performance, the BLEU method was quite good at picking up directly similar phrases to the bank, and in terms of score thresholds was quite clear when it had detected one. However, when it missed a red flag, it missed hard. We determined positive match score thresholds to be at or above around .66-.68, and misses to be usually less than .05.

The second methodology was sentiment analysis using SpaCy, a language model designed for projects like this. The idea was to compare the overall meanings of tokens, and not just how similar they were. This also gave quite good results, and we used the same reference phrase bank. SpaCy was able to pick up more of the edge cases that BLEU missed, however, had less predictable performance than BLEU and missed more of the obvious red flags.

For this reason, we landed on a final methodology of using weighted BLEU and SpaCy scores to generate an average score using tested thresholds. This accomplishes two things. Firstly, it allows us to establish a number that we can confidently say "if the score is over this threshold, it contains a red flag". Secondly, it allows us to normalize the score to a percentile. It is important to note that a BLEU score is *not* a confidence interval, i.e a score of .72 does *not* indicate a 72% confidence that a match is detected. By establishing our own confidence interval, we can normalize the generated scores to a percentile. To see this in action, see the below snippet:

```
#Weighted average of BLEU and Semantic scores
#Adjusted composite score calculation with new thresholds
final_scores = {}
threshold = 0.7  #Example threshold for high semantic similarity
for sentence in sentences:
    adjusted_semantic_score = 1.0 if semantic_scores[sentence] > threshold else
    semantic_scores[sentence] / threshold
    final_score = 0.6 * bleu_scores[sentence] + 0.4 * adjusted_semantic_score
    final_scores[sentence] = final_score
```

There are a few points of consideration and further work:

- Firstly, it would be beneficial to separate red flags into categories, with each category weighted differently. The reason for this is that some red flags are more alarming than others. For example, phrases like "no questions asked" are hard to justify as being legitimate and not illicit, whereas phrases like "new in box" could easily be explained as maybe the seller just didn't need their legally obtained part. These 6 categories represent the majority of all encountered suspicious phrases, and by prioritizing some categories over others, better results could be seen when integrating with the risk score framework:

urgency_phrases = [

"must sell immediately",

"need to sell ASAP",

"urgent sale",

"available now",

"quick sale needed",

"act fast",

```
"limited time offer"
]

secrecy_phrases = [
"discreet transaction",
"confidential deal",
"private sale",
"no questions asked",
"discreet only"
]

trustworthiness_phrases = [
"no receipt",
"not registered",
"sold as is",
"no return policy",
"no warranty"
]

financial_risks_phrases = [
"cash only",
"wire transfer required",
"full payment upfront",
"non-negotiable price",
"no refunds"
]
```

```
pressure_tactics_phrases = [
"act now",
"limited time only",
"before it’s gone",
"exclusive deal for you",
"once in a lifetime offer"
]

unrealistic_promises_phrases = [
"perfect condition",
"100% guaranteed",
"best deal ever",
"lowest price on the market",
"priced to sell fast"
]
```

- As another point of improvement, the METEOR metric of phrase analysis could be used. Our work was focusing on scaling horizontally, by trying various methods of red flag analysis, and because the principle of this is similar to BLEU, we did not pursue it. However, testing could be done against BLEU to see if METEOR gave better or more consistent results.
- The tokenization of the post body poses issues if the punctuation of the original post is irregular. Issues could be encountered if thread actors deliberately used periods to break up sentences, causing false negative red flag results. Consider the following:

  “No questions asked, must sell these new in box items today”

  vs.

“No ques.tions ask. ed must se. ll these new i. n box items today”

The second sentence would be tokenized as “No ques”, “tions ask.”, “ ed must se.”, “ ll these new i.”, “ n box items today”. When looked at individually, it is unreasonable to expect that even a perfectly tuned model could detect any sort of anomaly with each sentence, other than them being nonsensical. Potential remedies are numerous, with downsides to each:

  - Feeding the whole post, especially if it is a long post, would result in a low similarity score due to the noise even if there were a red flag or red flags present.
  - Stripping the whole post of punctuation (leaving spaces) and arbitrarily splitting at a certain amount of words would also potentially cut off red flags, but would be a better option.
  - Auto tokenization?

- While it sounds unlikely, the performance of red flag detection using preexisting LLMs such as GPT4 or others is superb. The issue is the large overhead of cost, as well as processing time. However, if looking for performance similar to an expert hand-reviewing a post, this is a good option.

Future Work

Being that this project had a time and work-hour constraint, there were several ideas that we as a group wanted to explore but did not have the availability to. This section is a catch-all for further improvements that wouldn’t have fit anywhere else.

- Image analysis as part of the risk score calculation shows great promise in early trials. Automated analysis of Craigslist posts was done by extracting all images in a post, then sending to an OpenAI API endpoint capable of image analysis. The LLM was prompted to detect abnormalities in the image, such as the image not matching the description, being sealed in the wrapper, security tags being attached, etc. Further work could be done by training a custom lightweight model purely to detect red flags in the image.
- Automating other parameters of the risk score formula.
  - See Research and Risk Score Formula.pdf

References

- Carver 2014

- ORC Report