A Hybrid Approach to Identifying Unknown Unknowns In Predictive Models

10/28/19

Presented by: Colin Vandenhof
Not all errors are created equal.

- In predictive models, high-confidence errors (i.e. unknown unknowns - UUs) are often more consequential than low-confidence errors.

- Why should we identify UUs?
  - Debugging the model
  - Preempting adversarial attacks
  - Model evaluation
Previous approaches to identifying UUs

- Two general approaches currently exist.

1. **Crowdsourcing**: candidates are proposed by workers

**Diagram:**
- Human: find/generate candidate, get true label
- Black box classifier: get model prediction
Previous approaches to identifying UUs

- Two general approaches currently exist.

1. **Crowdsourcing**: candidates are proposed by workers

   - “Beat the Machine”: Crowdsourcing task to submit webpages that will be misclassified by the model as hate-speech. Incentivized to find high confidence errors with bonuses (Attenberg et. al. 2015).
Previous approaches to identifying UUs

- Two general approaches currently exist.

2. **Algorithm**: candidates are selected algorithmically from a fixed set of instances.

- Human select candidate, get true label
- Test set with selection algorithm, black-box classifier
  - Select candidate, get model prediction
Previous approaches to identifying UUs

- Two general approaches currently exist.

2. **Algorithm**: candidates are selected algorithmically from a fixed test set

- Cluster all candidates (instances predicted with high-confidence) by their features and confidence scores.

- Candidates are selected from the most promising clusters based on their expected utility (Lakkaraju et al. 2017, Bansal et al. 2018).
Previous approaches to identifying UUs

- Two general approaches currently exist.

2. **Algorithm**: candidates are selected algorithmically from a fixed test set

   - Cluster all candidates (instances predicted with high-confidence) by their features and confidence scores.

   - Candidates are selected from the most promising clusters based on their expected utility (Lakkaraju et al. 2017, Bansal et al. 2018).

![Diagram illustrating high confidence instances and error (i.e. UU)]
Weaknesses

Crowdsourcing approach:

- Fails to explain the model’s behavior (i.e. how the model makes high confidence predictions). The model is a black-box to workers, so it is difficult to infer how to “beat” it.

Algorithmic approach:

- For models that are continually being adjusted, it may be inadequate to identify UUs from a fixed set.
- Fail to take advantage of human expertise.
Weaknesses

Crowdsourcing approach:
- Fails to explain the model’s behavior (i.e. how the model makes high confidence predictions). The model is a black-box to workers, so it is difficult to infer how to “beat” it.

Algorithmic approach:
- For models that are continually being adjusted, it may be inadequate to identify UUs from a fixed set.
- Fail to take advantage of human expertise.
Our hybrid approach

We design a crowdsourcing task called *Contradict the Machine*, in which decision rules can augment the ability of workers to generate UUs.
Our hybrid approach

**Phase 1:**
Decision Rule Learning
Explain how high-confidence decisions are made

- R1: feature_1 AND feature_2 => high-confidence pred c
- R2: feature_4 => high-confidence pred c
- R3: feature_6 AND feature_7 => high-confidence pred c
- R3: feature_7 AND feature_9 => high-confidence pred c

**Phase 2:**
Contradict the Machine
Search for UUs

- If not UU, modify the instance to contradict the rule
- get model prediction on modified instance
- select candidate, get covering rule

Explain how high-confidence decisions are made
Search for UUs
Phase 1: Decision rule learning

- We seek to learn a surrogate model that explains how the predictive model makes high-confidence predictions to the critical class $c$.

- This surrogate model is a set of decision rules of the form

  $\text{feature}_1 \text{ AND feature}_3 \text{ AND } \ldots \text{ AND feature}_n \Rightarrow \text{high-confidence } c \text{ prediction}$

  E.g. spam classifier

  “free” AND “buy” AND “now” $\Rightarrow$ high-confidence spam prediction

- Two desirable properties:
  - Interpretability: human can determine the when a rule applies to an instance
  - Decomposability: at most one rule applies to any instance
Phase 1: Decision rule learning

- We seek to learn a surrogate model that explains how the predictive model makes high-confidence predictions to the critical class \( c \).

- This surrogate model is a set of decision rules of the form

\[
\text{feature}_1 \text{ AND feature}_3 \text{ AND } \ldots \text{ AND feature}_n \Rightarrow \text{high-confidence c prediction}
\]

E.g. spam classifier

“free” AND “buy” AND “now” \( \Rightarrow \) high-confidence spam prediction

- Two desirable properties:
  - **Interpretability**: human can determine the when a rule applies to an instance
  - **Decomposability**: at most one rule applies to any instance
Phase 1: Decision rule learning

- We seek to learn a surrogate model that explains how the predictive model makes high-confidence predictions to the critical class $c$.

- This surrogate model is a set of decision rules of the form

  $$\text{feature}_1 \ \text{AND} \ \text{feature}_3 \ \text{AND} \ \ldots \ \text{AND} \ \text{feature}_n \Rightarrow \text{high-confidence} \ c \ \text{prediction}$$

  E.g. spam classifier

  “free” AND “buy” AND “now” => high-confidence spam prediction

- Two desirable properties:
  - *Interpretability*: human can determine the when a rule applies to an instance
  - *Decomposability*: at most one rule applies to any instance
Phase 1: Decision rule learning

- Data is discretized into instances predicted (1) or not predicted (0) to class c with high-confidence.

- A decision tree is generated via CART algorithm with modified splitting criterion.

- Every path of the decision tree from root to leaf is traversed. The rules correspond to all paths to a leaf with a class 1 majority.

```
"free" AND "claim" AND "cash" => high-confidence spam prediction
"free" AND NOT "ok" => high-confidence spam prediction
```
Phase 1: Decision rule learning

- Data is discretized into instances predicted (1) or not predicted (0) to class c with high-confidence.
- A decision tree is generated via CART algorithm with modified splitting criterion.
- Every path of the decision tree from root to leaf is traversed. The rules correspond to all paths to a leaf with a class 1 majority.

```
"free" AND "claim" AND "cash" => high-confidence spam prediction
"free" AND NOT "ok" => high-confidence spam prediction
```
Phase 1: Decision rule learning

- Data is discretized into instances predicted (1) or not predicted (0) to class c with high-confidence.
- A decision tree is generated via CART algorithm with modified splitting criterion.
- Every path of the decision tree from root to leaf is traversed. The rules correspond to all paths to a leaf with a class 1 majority.

```
"free" AND "claim" AND "cash" => high-confidence spam prediction
"free" AND NOT "ok" => high-confidence spam prediction
```
Phase 1: Decision rule learning

- Data is discretized into instances predicted (1) or not predicted (0) to class $c$ with high-confidence.
- A decision tree is generated via CART algorithm with modified splitting criterion.
- Every path of the decision tree from root to leaf is traversed. The rules correspond to all paths to a leaf with a class 1 majority.

```
"free" AND "claim" AND "cash" => high-confidence spam prediction
"free" AND NOT "ok" => high-confidence spam prediction
```
Phase 2: Contradict the Machine

- We use the decision rules to search for UUs via a crowdsourcing task called **Contradict the Machine (CTM)**.

- The worker is given a candidate (instance predicted with high confidence to $c$) and a rule that covers it.

- They can take one of three possible actions:
  - **identify**. Performed if the label is not $c$, since it is confirmed to be a UU.
  - **modify**. Otherwise, the worker is challenged to modify the instance such that its label changes, while ensuring that it is still covered by the rule. This makes a “contradictory” instance.
  - **reject**. Performed if the worker is unable to modify the instance.
Phase 2: Contradict the Machine

- We use the decision rules to search for UUs via a crowdsourcing task called **Contradict the Machine (CTM)**.

- The worker is given a candidate (instance predicted with high confidence to $c$) and a rule that covers it.

- They can take one of three possible actions:
  - **identify**. Performed if the label is not $c$, since it is confirmed to be a UU.
  - **modify**. Otherwise, the worker is challenged to modify the instance such that its label changes, while ensuring that it is still covered by the rule. This makes a “contradictory” instance.
  - **reject**. Performed if the worker is unable to modify the instance.
Phase 2: Contradict the Machine

- We use the decision rules to search for UUs via a crowdsourcing task called **Contradict the Machine (CTM).**

- The worker is given a candidate (instance predicted with high confidence to $c$) and a rule that covers it.

- They can take one of three possible actions:
  - **identify.** If the label is not $c$, it is confirmed to be a UU.
  - **modify.** Otherwise, the worker is challenged to modify the instance such that its label changes, while ensuring that it is still covered by the rule. This makes a “contradictory” instance.
  - **reject.** Performed if the worker is unable to modify the instance.
Phase 2: Contradict the Machine

- To sequentially select the instance (and covering rule) to next present to the worker, rules are treated like arms of a multi-armed bandit.

- Thompson sampling is used to trade off exploitation of the most promising rules with exploration.
Phase 2: Contradict the Machine

- To sequentially select the instance (and covering rule) to next present to the worker, rules are treated like arms of a multi-armed bandit.

- Thompson sampling is used to trade off exploitation of the most promising rules with exploration.
Experiments
Datasets

- We evaluate our method by conducting a user study on Amazon Mechanical Turk. We train classifiers on three datasets:

  1. **Rotten Tomatoes movie reviews**
     - Reviews labelled as negative or positive.
  
  2. **Amazon Food reviews**
     - Reviews labelled as negative (1-2 stars) or positive (4-5 stars).
  
  3. **SMS text spam**
     - Text labelled as non-spam or spam.
Datasets

- Following prior work, we induced bias in the training data to ensure that there were sufficient UUs to be discovered. This entailed:

1. Clustering the training data and removing data corresponding to a random cluster.

2. Biasing the class distribution by removing examples from the majority class (SMS text spam).
Datasets

- Following prior work, we induced bias in the training data to ensure that there were sufficient UUs to be discovered. This entailed:
  
  1. Clustering the training data and removing data corresponding to a random cluster.
  
  2. Biasing the class distribution by removing examples from the majority class (SMS text spam).
Crowdsourcing interface

instance

rule

modified text

three actions

identify modify reject

Original text:
a sun-drenched masterpiece, part parlor game, part psychological case study, part droll social satire.

Rules:
Include these words
- masterpiece

Exclude these words
- absorbing
- and
- around
- best
- comedies
- delivers
- enjoyable
- fun
- great
- heart
- human
- of
- perfectly
- performances
- refreshing
- solid
- still
- though
- urban
- who
- worth

Modified text:
a terrible film, part parlor game, part psychological case study, and all around boring.

Reset
User study

- The HIT was comprised of three sections:
  - **pre-study questionnaire** (demographics information)
  - **CTM tasks** (10 steps)
  - **post-study questionnaire** (TLX + questions about the difficulty of the task).

- Base payment of $0.50, plus action payments. The *identify* and *reject* costs were both set to $0.02, while *modify* cost was set to $0.20.

- Each classifier was evaluated over multiple HITs for a total of 300-500 steps.
User study

- The HIT was comprised of three sections:
  - **pre-study questionnaire** (demographics information)
  - **CTM tasks** (10 steps)
  - **post-study questionnaire** (TLX + questions about the difficulty of the task).

- Base payment of $0.50, plus action payments. The **identify** and **reject** costs were both set to $0.02, while **modify** cost was set to $0.20.

- Each classifier was evaluated over multiple HITs for a total of 300-500 steps.
User study

- The HIT was comprised of three sections:
  - pre-study questionnaire (demographics information)
  - CTM tasks (10 steps)
  - post-study questionnaire (TLX + questions about the difficulty of the task).

- Base payment of $0.50, plus action payments. The identify and reject costs were both set to $0.02, while modify cost was set to $0.20.

- Each classifier was evaluated over multiple HITs for a total of 300-500 steps.
We evaluated our approach (CTM) against several baselines:

- **UUB**: A re-implementation of the algorithm proposed by Lakkaraju et al.
- **CTM-NoRule**: A variant of CTM that does not present the worker with any rule that the modified instance must satisfy.
- **CTM-Random**: A variant of CTM that randomly selects instances to present to workers instead of the bandit algorithm.
Results
Cumulative utility

- At each step, the utility is calculated by the utility for identifying a UU (+1) or not (0), minus the cost of the action taken by the worker at that step.
Cumulative utility

- **CTM** performs better than **UUB** on all three datasets. The percentage increase in cumulative utility of CTM over UUB was 67.5, 32.1 and 68.5 respectively.
Cumulative utility

- Comparison of **CTM** with **CTM-NoRule** suggests that the rules are important, but their importance may vary between datasets, depending on the rule precision.

- **Movie reviews**: 60.7%
- **Food reviews**: 79.2%
- **SMS text spam**: 84.2%
Cumulative utility

- Comparison of CTM with CTM-Random suggests that the bandit query strategy may not be important.
Algorithm vs. worker contributions

- Breakdown of UUs discovered from the test set (i.e. algorithm proposed) and UUs generated by the worker (i.e. worker proposed).
- Both contributions are substantial, indicating the value of a hybrid approach.
UUs generated – common themes

- Changing the meaning of a word feature
  - E.g. SMS text spam: “free” in the sense of cost vs. “free” as in available

Include these words:
- free
- with

Exclude these words:
- call
- mobile
- reply
- www

Spook up your mob
with a Halloween
collection of a logo & pic
message plus a free eerie
tone, txt CARD SPOOK to
8007 zed
08701417012150p per
logo/pic

Do you think you’re
free to meet with me on
Tuesday?
UUs generated – common themes

- Manipulating context
  - E.g. modifying a review from calling the product “great” to saying that “indistinguishable people” think the product is “great”
  - E.g. SMS text spam: putting the entire spam text in quotes and complaining how much you dislike receiving such messages.

![Diagram showing examples of manipulating context]

Include these words:
- but
- great
- in
- thought

Exclude these words:
- awful
- bad
- bitter
- description
- nasty
- not
- off
- same
- should
- tasted
- what
- where

Example:
- I got hooked on Bigelow's Earl Gray tea. I thought I'd try Twinings version. Twinings is milder in flavor, but still very good, and a great value too.
- I got hooked on Bigelow's Earl Gray tea. I thought I'd try Twinings version. Twinings is milder in flavor, and I hated it, but I'm sure some indistinguishably people think it's great.
Summary

- This work proposes a hybrid approach to identifying UUs, in which candidates are generated by both the algorithm and human workers.

- To combine these approaches, we propose learning a set of decision rules that explain how high confidence predictions are made.

- We design a crowdsourcing task called Contradict the Machine, in which these decision rules can augment the ability of workers to generate UUs.

- Experimental results suggest that this method can outperform existing approaches.
Future directions

- Adapting interface to other data types
  - Tabular data
  - Longer text
- Adding mechanisms to take advantage of worker expertise
Future directions

▪ Adapting interface to other data types
  ▪ Tabular data
  ▪ Longer text

▪ Adding mechanisms to take advantage of worker expertise