RIPTIDE: Learning Violation Prediction Models from Boarding Activity Data

Hans Chalupsky
Information Sciences Institute
University of Southern California
Marina del Rey, CA U.S.A.
hans@isi.edu

Eduard Hovy
Language Technologies Institute
Carnegie Mellon University
Pittsburgh, PA U.S.A.
hovy@cmu.edu

Abstract—Part of the U.S. Coast Guard's mission is to monitor vessels and their operators for compliance with a large body of safety and fisheries regulations. Recently the Coast Guard has devised a system called OPTIDE, which aims at improving operations efficiency by ranking vessels via a risk score computed from current information and aggregated past boarding observations. Ships with higher risk should be preferentially boarded, since they have higher probability of being in violation of some regulation. To improve upon OPTIDE, we developed RIPTIDE which uses machine learning to automatically learn a more fine-grained and data-driven violation prediction and ranking model from past boarding activity data. The learning problem is challenging, since the data is very unbalanced (only about 20% of all boardings actually find some violation), it has significant sampling bias, and in general the signal for predicting violations is weak. Nevertheless, our best RIPTIDE model outperforms OPTIDE by up to 86% on a ranking experiment. The main reason for this improvement comes from being able to distinguish vessels in a more fine-grained manner, which allows RIPTIDE to make winning decisions more often, even if the underlying signal is very weak. A software package implementing RIPTIDE has been developed to allow the Coast Guard to experiment with the learned models and apply them to operational data.

Keywords: maritime law enforcement; fisheries; risk estimation; machine learning

I. INTRODUCTION

Many agencies monitor large-volume and distributed activities for violations. Since it is usually impossible to check every activity, agents have to select whom to inspect in a given time period. Over time, agents tend to develop criteria that improve the likelihood of finding violations. Usually these criteria are intuitive and not scientifically tested. In this paper we exemplify a methodology for testing criteria, showing by example that using a machine learning classifier can improve even a carefully crafted manually built selection procedure.

The United States Coast Guard (USCG) is tasked with a very broad range of missions including enforcement of safety and fisheries regulations, drug and illegal immigration interdiction, search and rescue, disaster response, port security, general homeland security and many others. Ever tightening operational budgets force the USCG to continually improve operational efficiency and do more with less.

A central mission of USCG District 1 headquartered in Boston is fisheries law enforcement, which involves monitoring of vessels and their operators for compliance with a large body of safety and fisheries regulations. To improve its operational efficiency in this area, USCG D1 has recently devised a system called OPTIDE, which ranks boardable ships via a risk score computed from current information and aggregated past observations. Ships with higher risk should be preferentially boarded, since they have higher probability of being in violation of some regulation. OPTIDE uses manually devised scoring rules developed by subject matter experts and significantly increased the number of violations found per boarding conducted. Nevertheless, the somewhat ad hoc scoring rules lacked a strong statistical foundation in the underlying data and led to some undesired characteristics of the overall risk distribution.

To improve upon OPTIDE, we developed RIPTIDE (which loosely stands for Rule Induction oPTIDE). Instead of manual rule formulation, RIPTIDE uses machine learning to automatically learn a more fine-grained and data-driven violation prediction and ranking model from past boarding activity data. The learning problem is challenging, since the data is very unbalanced (only about 20% of all boardings actually find violations), has significant sampling bias, and in general there are no strong "red flags" and the signal for predicting violations is weak. Nevertheless, our best RIPTIDE model using boosted decision trees outperforms OPTIDE by up to 86% on a ranking experiment. The main reason for this improvement comes from a finer-grained risk distribution, which allows RIPTIDE to make winning tradeoff decisions about which ships to board more often, even if the underlying violation signal for each individual vessel is not very strong.

We developed a software package implementing RIPTIDE that comes with several pre-configured models that make different tradeoffs on which features they require vs. the maximum performance they can achieve. The software also allows retraining of models and experimenting with new features, which is important to allow the models to stay current with a constantly evolving and adapting population of vessel operators. The software has been delivered to the Coast Guard who are evaluating it for operational use.

II. CURRENT APPROACH: OPTIDE

To motivate our approach taken with RIPTIDE, we first describe OPTIDE which is a relatively recent system developed by USCG’s District 1 (D1) to improve their overall operational efficiency. D1 spans the Atlantic coast line from Maine all the way to New Jersey. The New England area of responsibility alone accounts for about 45% of USCG D1’s mission with over 4,000 federally licensed commercial vessels operating in this region. The main commercial activity in this area is trapping and fishing for lobster, scallops, cod, haddock, flounders, bluefish, black sea bass, tuna and various other species. The Coast Guard therefore needs to perform continuous law enforcement to (1) enforce fishery regulations such as monitoring of quotas, fishing seasons, allowed equipment such as net types, and whether vessel owners are adequately licensed for the type of fish they are catching, and (2) enforce safety regulations and safe operations to minimize loss of life and equipment. Vessels are monitored at a first level by air stations such as helicopters that can observe a vessel’s location and activity, and in more detail by Coast Guard cutters in close proximity to a vessel. The most important inspection and law enforcement tool available is the boarding of a vessel, where detailed information about the catch, fishing and safety equipment and licensing can be obtained. Unfortunately, boarding a vessel is also a resource-intensive operation that involves potentially significant travel time to reach the vessel and then the time it takes to inspect it which can take from 15 minutes to many hours depending on the size and type of the ship. Given these requirements and limited resources, the Coast Guard plans its daily operations trying to satisfy a number of different and potentially conflicting goals such as maximizing the number of violations found, maximizing the deterrent for future violations while still providing balanced policing for the whole area.

A recently developed tool for this task is OPTIDE, which is a rule that computes a risk score for a vessel based on current observations and aggregated historical data observed during prior boardings. OPTIDE considers a number of features and for each of them observes or computes a value. Each feature value has a risk score associated with it. The sum of these risks is the total OPTIDE risk.

Table 1 shows a notional computation of OPTIDE risk from a number of different features. Since this rule is actively used by the Coast Guard, feature values and risk scores are simply notional and disguised, but the table illustrates the general principle. Features consider particular properties of a boat, violations found in the past, temporal information, etc. The combined risk then is a simple weighted sum:

\[ \text{Risk} = \sum R_i \cdot C_i \]

Each risk score \( R_i \) serves as a weight which is multiplied with 1 if condition \( C_i \) is satisfied, and 0 otherwise. If the combined risk is greater than a threshold, the Coast Guard considers the vessel for boarding.

The OPTIDE rule was devised manually by Coast Guard officers, based on their expertise and intuition. Its introduction significantly increased the average number of violations found per boarding conducted, and it also led to a more balanced monitoring of vessels in their jurisdiction. Nevertheless, the rule is \emph{ad hoc} and lacks a strong statistical foundation in the underlying data. This led to the question whether it could be further improved using more principled data analytics which motivated the development of RIPTIDE. It also is the reason for some undesirable characteristics of the risk distribution which we discuss below.

III. NEW APPROACH: RIPTIDE

To improve upon OPTIDE we developed RIPTIDE which uses machine learning to automatically find regularities in past boarding activity data and encodes those in a model that can then be used to rank new boarding activities. This is the decision problem a commanding officer faces every day.

We modeled this as a classification problem. Classification algorithms are the most common and well-studied machine learning methods in Artificial Intelligence. A classifier takes a data instance and a previously learned model as inputs and then makes a prediction on which class the instance belongs to. For our application, we model a particular vessel by a set of values for attributes or features (similar to the approach taken by OPTIDE), and then ask the classifier to predict either “board” or “don’t board” based on those feature values.

Machine learning is built upon two core principles: data representation and generalization. First, every data instance has to be represented in some computer-understandable form. This is generally done by engineering a set of features or attribute/value pairs that carry some relevant information and that can either be directly observed or computed from the data. For example, we might use a “hull-type” feature that would encode the material of a vessel’s hull (e.g., “wood”, “steel”, etc.). In the generalization phase, the classifier uses many data instances for which the class is known as training data, and then tries to find regularities in that data that might allow it to predict the class of a new data instance. There are many different data representation schemes and learning algorithms that can be used for such purposes. For example, see [1,2,3] for an overview.

As the RIPTIDE classifier we chose a boosted decision tree algorithm, which is a good general-purpose choice for
problems with a small to medium number of features. One of the advantages of decision trees is that the learned models are essentially large ‘if-then-else’ statements that are to some extent understandable by humans. We thought this to be a useful property when comparing to a rule-based approach such as OPTIDE, since humans might want to have some reason why they should trust such a model. (Other popular learning methods such as support vector machines or neural nets, for example, produce completely opaque models that can only be judged by their input/output behavior.) Here is a fragment of an example, produce completely opaque models that can only be trusted such as OPTIDE, since humans might want to have some reason why they should trust such a model. (Other popular learning methods such as support vector machines or neural nets, for example, produce completely opaque models that can only be judged by their input/output behavior.) Here is a fragment of a decision tree learned by RIPTIDE (again, feature names and values have been disguised):

```
prior_feature6 <= 1.33
   | optide_feature2 = Value2.1: 0.0 (3706.29/861.91)
   | optide_feature2 = Value2.2: 0.0 (630.0/91.0)
   | optide_feature2 = Value2.3
   | type_feature3 = Type3.1
   |   | prior_feature4 <= 2: 0.0 (392.0/66.0)
   |   | prior_feature4 > 2
   |   | temporal_feature5 = N5.1: 1.0 (5.0/1.0) (*)
   |   | temporal_feature5 = N5.2: 0.0 (4.0/1.0)
   |   | temporal_feature5 = N5.3: 0.0 (6.0/2.0)
   |   | temporal_feature5 = N5.4
   |   |   | prior_feature6 <= 0.91: 0.0 (5.0)
   |   |   | prior_feature6 > 0.91: 1.0 (2.0)
   | type_feature3 = Type3.2: 0.0 (244.0/24.0)
   type_feature3 = Type3.3
   type_feature7 = Type7.1: 0.0 (16.0/2.0)
   type_feature7 = Type7.2
   | prior_feature6 <= 0.31: 1.0 (5.0/2.0)
   | prior_feature6 > 0.31: 0.0 (3.0)
   | type_feature7 = Type7.3: 1.0 (1.0)
   | type_feature7 = Type7.4: 0.0 (0.0)
```

The leaf marked with (*) represents the following rule:

```
IF  prior_feature6 <= 1.33
   & optide_feature2 = Value2.3
   & type_feature3 = Type3.1
   & prior_feature4 > 2
   & temporal_feature5 = N5.1
THEN predict BOARD
```

Each line represents one choice step, including a test (“feature2 = Value2.1”) and trustworthiness scores at that point. If the test is true then the algorithm proceeds to the next line that is one step indented; if false, to the next line that is not. At the end, the algorithm selects BOARD or NO-BOARD depending on the final criteria.

Classification performance can generally be improved by combining multiple classifiers that were trained using different algorithms, features, sections of the data and any combination thereof. One such strategy is called boosting, where instead of learning just a single decision tree we learn multiple trees on different subsets of the training data. An algorithm such as AdaBoost [4] for Adaptive Boosting then learns weights for combining the results of those individual decision trees into an overall boosted decision tree. AdaBoost focuses the learning of subsequent classifiers on instances that have been misclassified by previous classifiers. For our currently best-performing Model 58, boosting improves performance on a boarding tradeoff task by ~25%.

A. Boarding Activity Data

Our analysis and model development is based on two data sets that were provided to us by the Coast Guard. One covers activities from 2002 to 12/20/2011 and was used to train a variety of different RIPTIDE models. The second one covers activities from 12/21/2011 to 09/30/2012 and was used as a held-out test set, not visible during learning, to evaluate the models and compare their performance to OPTIDE.

The data was extracted by USCG personnel from the Coast Guard’s Marine Information for Safety and Law Enforcement (MISLE) database. The data records about 24,000 activities of which 46.6% are vessel boardings and the rest are sightings. Each data entry has an activity ID, an anonymized vessel ID, and 24 different raw features that describe time and location of an activity, vessel attributes such as type and length, type of fishery, vessel activity, length of the boarding activity, as well as details about any violations found. Sights did not have any violation information associated with them; hence we ignored them and focused only on boarding activities.

The data is significantly unbalanced. Only 20.5% of all boarding activities actually found one or more violations. Moreover, most of the violations found were safety violations and only 3.4% of all boardings find one or more fisheries violations, such as a missing or insufficient permit. When a violation is found, there usually is more than one with an average of two violations per qualifying boarding.

Besides being unbalanced, the data also exhibits significant temporal variation. Starting in about 2006, the numbers of violations reported has generally been going down with only about 14% of boardings finding some violation in the 2012 segment of the data (compared to the 20.5% average). There is also strong seasonal variation as shown in Figure 1. Particularly for the second half of the data, there seem to be fairly consistent dips in the summer and rises in the winter and spring. Some of these dynamics can be attributed to more vigilant enforcement at the beginning of winter to ensure vessel operators are prepared for the upcoming inclement weather. Other reasons might be more recreational boating activities during the summer months which might divert some of the Coast Guard’s resources.

Finally, it should be noted that the data is not a random sample of vessels and their violation status. Rather, commanding officers select vessels to board based on their knowledge and experience and the specifics of a particular situation. Moreover, since the introduction of OPTIDE in 2011, many of these decisions were guided by OPTIDE risk scores. For this reason, training and evaluating learning algorithms on this data is somewhat challenging, since we are evaluating on an already filtered and somewhat optimized population.

---

2 See for example http://cgmix.uscg.mil/
B. Learning RIPTIDE Models

The precise classification task we are trying to solve is the following: given a set of currently observable features about a vessel and its known history as recorded in the MISLE database, estimate the probability that some violation will be found if the vessel were boarded. This is different than a simple “board/don’t board” decision, since the latter will always have to involve some threshold that is generally context dependent and often difficult to pick. Instead, by providing a probability estimate, we can rank vessels relative to each other which allows one to make informed local decisions independent of some arbitrary threshold.

Building a classifier involves deciding on a set of features to represent the data, learning a model based on those features, and then evaluating that model on held-out test data. This is generally an iterated process of trial-and-error guided by human intuition. We started with the features considered by OPTIDE and augmented them with additional features that seemed promising from our initial data analysis, for example, encoding seasons or the current month given the seasonal variation shown in Figure 1, or considering features that had higher than average correlation with detected violations.

Another consideration is how to represent features, which matters particularly for aggregated historic information. For example, we can represent violations found in the past as total counts, total counts per boarding, total count per time period, use some exponential decay to discount violations that are longer ago, etc. Moreover, we might consider that some violation was found only during a boarding instead of the total count, separate out violation types (fishery or safety), etc. For attributes with large numbers of discrete values such as a vessel’s type (which has 38 different possible values), we optionally reduced the value set to the most frequent ones to reduce the dimensionality of the learning problem. Finally, given that there are defined zones such as state and federal waters with different associated permits, we also experimented with some encoding of lat/long location information such as the distance to the closest coast.

This large space of features and their possible combinations was then explored manually in a semi-principled hill-climbing fashion. From an initial feature set based on OPTIDE, we added and removed features one-by-one, observing their impact on model performance and keeping the winning features. We used the Weka machine learning toolkit [3] for our feature space exploration, starting with J48 (a Java implementation of Quinlan’s C4.5 algorithm [5]) and later on an Ada-boosted decision tree with the default decision stump as its base classifier, which generally performed better than a boosted version of J48.

There are 13 main violation classes in the data (which we simply separated into fishery and safety violations), with about 70 subtypes. Ideally, we would learn to predict the exact class of violation or at least separate between fishery and safety violations (especially since some of these violations are more serious than others). However, the sparsity of the data, particularly for fishery violations, did not allow us to do that with any reasonable performance. Therefore, for all our models we only learned a single violation/no-violation class.

For each considered combination of features we ran tenfold cross-validation on a 70/30% train/test split of the 2002-2011 training data to select the best model of this type. We kept the model from the best-performing fold as the one to compare against OPTIDE on the 2012 held-out data. Performance was measured with a tradeoff experiment of bucket size 20 described in more detail below.

Our feature space exploration led us to three different models that make slightly different tradeoffs. The best-performing Model 58 uses seven different features. Three of them are derivates of OPTIDE features. Two of those have very low weights assigned in the boosted combination, but the...
third one is actually the most important in the model with almost 60% weight. It is also the most important feature for the other two models and validates the OPTIDE approach. The second most important feature of this model is based on the vessel’s location and the third one on its (simplified) type. The model also takes into account some temporal and seasonal information, however, its performance is not affected much if those features are eliminated. We are intentionally vague about the exact nature of the various features due to their potential application in Coast Guard operations.

The second-best model is Model 57 which is identical to Model 58 except for the location feature. This means the feature values for this model can be computed without having to know the location of a vessel. Finally, a third Model 48 uses a fine-grained representation of a vessel’s activity as its third-most important feature which usually can only be observed in close proximity.

IV. EVALUATION

To evaluate the RIPTIDE models described above, we used the held-out 2012 data of 1002 boarding activities. When we use a classifier such as RIPTIDE, we generally have to pick a threshold which we can estimate from the training data. If the probability estimate is above the threshold, we will decide to board a vessel, if it is below, we will not board. However, classifier performance usually depends heavily on where we set the threshold. Let TP be the number of true positives we get (that is cases where the classifier says “board” and we in fact found a violation); the remaining cases where the classifier says “board” are the false positives (FP). A standard evaluation metric for classifiers uses recall R (the percentage of boardings with some violation detected), precision P = TP/(TP+FP) measuring the fraction of true decisions, and their harmonic mean, known as the F1 value, namely F1 = 2*P*R/(P+R). Picking a low probability threshold will give high recall but low precision, conversely, a high threshold will give high precision but low recall. Every threshold therefore represents a tradeoff between TP and FP, and what is acceptable depends on external factors such as task objectives and resources. Using a generic point such as maximum recall or F1 value will generally not give the best compromise in practical applications.

One way to compare classifiers without using a threshold is to plot ROC curves (ROC stands for Receiver Operating Characteristic and was a method developed in WWII to evaluate human analysis of radar signals). In an ROC curve we plot for each possible threshold point the true positive rate (or recall), that is the ratio of good predictions we made, over the false-positive rate, that is the ratio of false predictions we made. The curve shows a tradeoff space where we see how many more false positives one must accept to get additional true positives.

We can use the area under the ROC curve to compare different classifiers, where a higher area under the curve generally means a better classifier. Figure 2 shows a comparison of ROC curves for OPTIDE and our best Model 58 for the held-out test data covering the year 2012. Both models have more or less identical area under the curve (AUC) of about 0.65, which shows that they are doing better than random choice (the dotted line with an AUC of 0.5), but not very significantly so. This indicates that there is not a very strong signal in the data to begin with. Model 58 is doing significantly better for picking up higher risk boardings (the bump at the beginning of the curve), but then loses that advantage towards lower-risk boardings. It also is much more fine-grained then OPTIDE which is a feature we discuss in more detail below.

One of the problems with the current formulation of OPTIDE is that for most of the boardings, the risk distribution is very flat which can be seen in the long straight sections of the OPTIDE ROC curve. About 84% of all boardings fall in a very narrow band of risks close to the threshold level. That means a large number of ships are basically indistinguishable. In fact, from our understanding of the data and observable parameters, there are no standout red flags that clearly indicate that a ship might be in violation of some regulation. Even at the highest risk score, only one third of boardings having this risk actually led to a violation being found. This means we cannot assign a strong meaning to any OPTIDE risk category.

![Figure 2. ROC curves for OPTIDE and Model 58 for the held-out test data in 2012](image_url)
that was selected by some combination of officer in tuition, sample of ships to board; rather it already presents a filtered set than the 2002–2011 data. For example, the average rate for years which would explain that it fits the 2012 data better. One reason for this might be that OPTIDE was applied more almost 14% improvement of AUC over the 2002–2011 data. OPTIDE performs significantly better on the 2012 data with an average 20.5% on average for 2002–2011. Also, boardings that found some violation is only about 14% in 2012, RIPTIDE significantly outperforms OPTIDE with its large bucket size of 10, the improvement is still a good 38%. This 45% chance to pick a winner, and even for a more realistic over OPTIDE for a bucket size of 20 where we have an almost different characteristics.

Table 2 shows the results of our evaluation experiments. The top portion shows standard AUC and Max-F1 metrics, and all models perform fairly similarly. In the lower portion, we show results on ranking experiments with bucket sizes ranging from 2-50. We find that our best model improves up to 76% over OPTIDE for a bucket size of 20 where we have an almost 45% chance to pick a winner, and even for a more realistic bucket size of 10, the improvement is still a good 38%. This shows that the more fine-grained decision rule implemented by RIPTIDE significantly outperforms OPTIDE with its large sections of undifferentiated risk.

Table 2: Performance for OPTIDE and RIPTIDE models

<table>
<thead>
<tr>
<th>Bucket Size</th>
<th>N-Threshold</th>
<th>Max-F1</th>
<th>AUC</th>
<th>Random</th>
<th>OPTIDE</th>
<th>Model 57</th>
<th>Model 48</th>
<th>Model 58</th>
<th>Model 58 vs. OPTIDE</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>0.135</td>
<td>0.210</td>
<td>0.217</td>
<td>0.236</td>
<td>0.243</td>
<td>+15.9%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>0.135</td>
<td>0.237</td>
<td>0.279</td>
<td>0.311</td>
<td>0.328</td>
<td>+38.5%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>0.135</td>
<td>0.244</td>
<td>0.328</td>
<td>0.364</td>
<td>0.393</td>
<td>+60.9%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>0.135</td>
<td>0.251</td>
<td>0.363</td>
<td>0.403</td>
<td>0.443</td>
<td>+76.4%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>0.134</td>
<td>0.281</td>
<td>0.399</td>
<td>0.440</td>
<td>0.484</td>
<td>+85.1%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>0.135</td>
<td>0.276</td>
<td>0.422</td>
<td>0.466</td>
<td>0.516</td>
<td>+86.8%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>35</td>
<td>0.135</td>
<td>0.290</td>
<td>0.447</td>
<td>0.488</td>
<td>0.542</td>
<td>+86.6%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>40</td>
<td>0.134</td>
<td>0.307</td>
<td>0.464</td>
<td>0.505</td>
<td>0.567</td>
<td>+84.7%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>50</td>
<td>0.137</td>
<td>0.336</td>
<td>0.492</td>
<td>0.542</td>
<td>0.601</td>
<td>+78.9%</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

V. DISCUSSION

As noted before, the 2012 data looks significantly different than the 2002–2011 data. For example, the average rate for boardings that found some violation is only about 14% in 2012, while it is about 20.5% on average for 2002–2011. Also, OPTIDE performs significantly better on the 2012 data with an almost 14% improvement of AUC over the 2002–2011 data. One reason for this might be that OPTIDE was applied more extensively to choose ships to board in 2012 compared to prior years which would explain that it fits the 2012 data better. Another issue with the data is that it is not an unbiased random sample of ships to board; rather it already presents a filtered set that was selected by some combination of officer intuition, mission objectives and/or OPTIDE. Still, Model 58 generalizes quite well to the 2012 data despite its significantly different characteristics.

In our ranking experiment, we sample with replacement, that is, once we pick a winner from a sampled bucket, we throw it back into the pool which means it could get chosen again in a subsequent trial. In reality, however, boarding a ship that had some violations will change that ship’s violation profile for future boardings, for example, safety violations will get fixed; therefore, using the RIPTIDE model would potentially lead to fewer violations being found over time. Only after some time has passed will new violations occur to replace the old ones that were addressed. Factoring this in would have made the experiment more realistic but also significantly more complicated. Despite the simplification, the relative comparison between OPTIDE and Model 58 still holds.

We have developed and packaged up a small RIPTIDE software suite that can be used to classify and rank potential boardings based on the models described above. It can also be used to retrain models if necessary. RIPTIDE builds upon the Weka toolkit [3] and adds a number of convenience methods for data translation and various other tasks. RIPTIDE is purely Java-based and can be run on all common OS platforms.

To use an approach like RIPTIDE in practice will require the users to retrain the machine learning models at regular intervals, maybe on a yearly basis, to ensure that any significant changes in behavior will be factored in. This would be very easy to do with the RIPTIDE software, as long as the basic set of features to consider remains the same or similar. The actual implementation of RIPTIDE is experimentally underway by the USCG.

ACKNOWLEDGMENTS

This research was made possible by grants from the U.S. Coast Guard District 1 and the U.S. Department of Homeland Security. The statements made herein are solely the responsibility of the authors. Many thanks to LCDR Ryan Hamel and LT Ryan Kowalske for their patience and support, and to Alisa Matlin and Bobby DeMarco and other members from the CCICADA Research Center for their help with the data and valuable comments and suggestions.

REFERENCES