Learning High Precision Rules to Make Predictions of Morbidities in Discharge Summaries

Ted Pedersen, PhD

University of Minnesota, Duluth, MN, USA

Abstract

The Duluth entries in the 2008 I2B2 Obesity Challenge used supervised machine learning techniques that relied on bag of words unigram features found in discharge summaries to predict if a patient is obese or suffers from any of 15 related co–morbidities. We found that the RIPPER rule learning algorithm created high precision models that exceed the mean precision of all the participating systems by a significant degree. It also discovers simple and informative rules that allow us to better understand the domain. However, no supervised learning algorithm that we experimented with was able to perform well on the minority judgments which make up less than 1% of the total training and test data.

Introduction

The I2B2 Challenge for 2008 was to predict if a patient suffered from obesity or any of 15 other co-morbidities under two different problem definitions. In the textual task, the prediction was based strictly on information found in the patient's discharge summary, and in the intuitive task, expert judgments were combined with the content of the discharge summary.

The Challenge organizers provided manually annotated training data which consisted of 730 discharge summaries which had each been annotated with judgements for the two different tasks. Given this training data, we decided to approach the Challenge as a problem in supervised learning. Our overall method was to learn a model for each morbidity from the training data, and then use that model to predict the morbidities present in 507 discharge summaries that made up a held–out test set.

We relied on textual information from the discharge summaries, where any single word (unigram) that occurred more than one time in the training data was considered a feature. The position of the unigram in the discharge summary was not considered, which means our feature set corresponds fairly closely to the traditional bag of words often used in text classification problems. While our method was purely textual, we applied it to both the textual and intuitive tasks, hypothesizing that expert intuitions are at least in part triggered by nuanced textual data they encounter while reviewing the discharge summaries.

Challenge Data

The training data consisted of 730 discharge summaries that were manually annotated with up to 16 judgments indicating the presence of obesity and its co-morbidities according to both the textual and intuitive task guidelines. The test data consisted of 507 discharge summaries that were also manually annotated, but those annotations were used for evaluation and withheld from participants until after the competition.

The textual task specified that a system predict one of four judgments for each of the 16 morbidities for a discharge summary: *Yes* (the presence of the morbidity in the patient is specifically mentioned), *No* (the absence of the morbidity is specifically mentioned), *Questionable* (the morbidity is mentioned but it is uncertain if it affects the patient), and *Unmentioned* (there is no mention of the morbidity).

The intuitive task specified that a system output one of three judgments for each morbidity: *Yes* (the patient suffers from the morbidity), *No* (the patient does not suffer from the morbidity), and *Questionable* (the patient might suffer from the morbidity).

The distribution of the judgments in the intuitive and textual tasks for both the training and test data are shown in Tables 1 and 2. Participants knew the distribution in the training data but not the test data (since the annotations were withheld until after the evaluation period). However, we assumed that the training data was representative of the test data, and this proved to be the case.

These tables show that the training and test data is very heavily skewed towards two dominant judgments,

Table 1: **Intuitive Task Judgment Distributions**

Judgment	Training 730 patients		Test 507 patients	
No	7,362	(.69)	5,100	(.69)
Yes	3,267	(.31)	2,285	(.31)
Questionable	26	(.00)	14	(.00)
Totals	10,655	(1.00)	7,399	(1.00)

Table 2: **Textual Task Judgment Distributions**

Judgment	Training 730 patients		Test 507 patients	
Unmentioned	8,296	(.71)	5,770	(.72)
Yes	3,208	(.28)	2,192	(.27)
No	87	(.01)	65	(.01)
Questionable	39	(.00)	17	(.00)
Totals	11,630	(1.00)	8,044	(1.00)

which account for more than 99% of the total judgments in both the intuitive and textual task. The potential effect of this on supervised learning can be quickly assessed by looking at the number of training examples available per morbidity. For example, in the intuitive task, the average number of training examples per morbidity for the judgment *No* is 460, for *Yes* is 204, and for *Questionable* is 2. In the textual task, the average number of training examples per morbidity for *Undecided* is 517, for *Yes* is 201, for *No* is 5, and for *Questionable* is 2.

We were also concerned that a minority judgment observed in the training data for a particular morbidity might not be observed at all in the test data, or that a minority judgment found in the test data might not be observed in the training data. We had these concerns because the minority judgments in the training data were so small in number, and somewhat unevenly distributed across the morbidities. In the intuitive task, 6 of the morbidities had no *Questionable* examples, and in the textual task 7 morbidities had no *Questionable* examples, and 6 morbidities had no *Unmentioned* examples. While this is a natural consequence of the nature of the data, it poses concerns for supervised learning, where it is essential that the training data be representative of the test data.

In retrospect, there were only a few minor differences between the training and test data with respect to the minority judgments. In the training examples for the intuitive task there were 26 *Questionable* judgments applied to 10 different morbidities, while in the test data there were 14 *Questionable* judgments that occurred in just 6 of those 10 morbidities. In the textual task training examples there were 39 *Questionable* judgments that occurred in 9 different morbidities, while in the evaluation data there were 17 *Questionable* judgments that occurred in those same 9 morbidities. In the textual training examples there were 87 *No* judgments that occurred in 10 different morbidities, while the evaluation data had 65 *No* judgments which occurred in 9 of those 10 morbidities. Thus, there were training examples provided that were essentially irrelevant to the test data (since the evaluation data did not contain some of those judgments). Fortunately there were no judgments for any morbidities in the test data that were not also included in the training data. This would be a far more problematic situation, since the correct judgment in that case would never have been seen in any form in the training data, and the learned model could not know about it.

Based on all of these considerations, we concluded that it would not be feasible for a supervised algorithm to effectively learn the minority judgments (*Questionable* in the intuitive task, and *No* and *Questionable* in the textual task) and there would be a risk of fragmenting the learning process by forcing it to attempt to learn in cases where the number of training examples is quite small. As such we decided to exclude the minority judgments from the training data, and proceed as if this was a two class problem in both the intuitive and textual tasks.

This was a calculated gamble, but at the time we were making this decision we did not know the distribution of the test data. Thus, we elected to take a fairly cautious approach, and focus on learning high precision models for the dominant judgments.

The most obvious drawback of this decision was for our Macro Recall scores. This measure is based on the average of the recall for each judgment, and by excluding certain judgements we would guarantee recall of 0 in those cases, which would drive down the macro score significantly since the number of judgments is small. For the intuitive task, we would get 0 recall for 1 of 3 judgments, and 0 recall for 2 of 4 judgments in the textual task.

Supervised Learning Methods

We took a textual approach to this problem, and relied on lexical features found in the discharge summaries in the training data to represent the training data and build models for classification. No external sources of information were used, and the features were identified with the Ngram Statistics Package[1]. During that process we converted all text to lower case, discarded punctuation, numeric data, and single character strings, and defined words as being space separated strings that were not included among the approximately 200 stop words found in the SMART stop list. We experimented with unigram, bigram, trigram, and 4–gram

features, but found that unigrams that occurred more than one time in the training data resulted in significantly more accurate models during cross–validation studies with the training data. There are approximately 9,000 unigram features per morbidity, although they vary somewhat with each morbidity since some annotations were withheld from the training data due to low inter–annotator agreement.

As a part of our studies to determine which features to use, we also evaluated a wide range of machine learning algorithms. After a round of preliminary evaluation using many of algorithm supported in the Weka Machine Learning Toolkit[2], we focused on the following five: The RIPPER rule learning algorithm[3] (JRip in Weka), the C4.5 decision tree learner[4] (J48 in Weka), a Support Vector Machine (SMO in Weka), a Voted Perceptron (VotedPerceptron in Weka), and a Naive Bayesian classifier (NaiveBayes in Weka). These represent a cross section of techniques: JRip and J48 discover rules that cover or partition the training examples while SMO, the Voted Percepton, and Naive Bayes are (to varying degrees) fitting the training examples to an underlying statistical or mathematical model.

Somewhat to our surprise, JRip and J48 performed significantly better than the other supervised learning algorithms. We had expected a somewhat closer competition between these methods since Support Vector Machines and Naive Bayesian Classifiers have a long history of success in text classification problems, and Voted Perceptrons have been applied to a range of language problems in recent years. However, it may be that the volume of training data was not sufficient to reliably fit the data to the models that these methods rely on.

In addition to being more accurate, JRip and J48 both learn rules that are easy to understand and provide informative feedback about the problem. These methods are similar in that both attempt to learn rules that explain or account for as much of the training data as possible, without over–fitting and simply memorizing the training examples. However, they take opposite approaches; JRip is a bottom–up method learns rules that cover all the examples for each possible judgment, while J48 is a top–down algorithm that partitions the training data into relatively pure subsets.

JRip (RIPPER) proceeds by treating all the examples of a particular judgment in the training data as a class, and finding a set of rules that cover all the members of that class. Thereafter it proceeds to the next class and does the same, repeating this until all classes have been covered. At no point does it consider all of the training examples as a whole (which is why we refer to it as bottom–up). J48 (C4.5) takes a top–down divide–and–conquer approach. It proceeds recursively by selecting a single feature that best divides the training examples into classes, where (in the end) each class is made up of examples of only one judgment (or as nearly so as possible). The first feature selected is the root of the decision tree, and the subsequent features form branches that eventually cover or explain some subset of the training examples.

Like nearly all machine learning algorithms, both JRip and J48 must balance over–fitting with generalization. If the model that is learned too closely characterizes the training examples, then the model will not fare well on new data. Both JRip and J48 engage in pruning of their rules to allow for this generalization. Usually this pruning is based on the number of training examples that are covered or explained by a particular rule or feature, so the rules or features that cover or explain the small minority judgments are most likely to be pruned away, leaving the resulting model unable to classify them.

JRip and J48 search for rules and features in a greedy fashion, making them vulnerable to locally–optimal choices early on that do not lead to overall globally–optimal results. Given this we felt that the meta–learner AdaBoost[5] was appropriate to use with both JRip and J48. AdaBoost proceeds iteratively through the training data, and re–applies the learning algorithm (either JRip or J48) on training examples weighted based on how easy or hard they have proven for previous iterations of the classifier to get correct. While we did not expect AdaBoost to result in hugely significant improvements to our results, we felt that it would help avoid any obvious problems caused by poor initial choices by either JRip or J48.

Each participating team was allowed to submit three sets of results for each task. We found that JRip with and without AdaBoost and J48 with AdaBoost were the most accurate of our methods, and so those systems were used to create our submitted results. The only difference between the intuitive and textual task systems was the data upon which they trained. While they were trained on the same set of discharge summaries, the underlying annotations differ between the tasks. For each of our three learning algorithms, we learned an intuitive and a textual classifier for each of the 16 morbidities, giving us a total of $3 \times 2 \times 16 = 96$ classifiers. Each of these was applied to the test data, and the output was submitted to the appropriate task of the I2B2 Challenge without any manual review or modification.

Challenge Results and Discussion

The systems that participated in the Challenge were scored on precision, recall, and the F-score using

Table 3: **Performance % (Intuitive Task):** training data limited to top 2 judgments, results of our official submissions plus post–evaluation studies

	Precision		Recall		F-Score	
Method	mic	mac	mic	mac	mic	mac
Mean	**91**	**78**	**90**	**60**	**90**	**60**
JRip	*93*	*95*	*93*	*60*	*93*	*61*
JRip-AB	*93*	*95*	*93*	*60*	*93*	*61*
J48-AB	*92*	*95*	*92*	*60*	*92*	*60*
J48	90	93	90	59	90	59
SMO	88	92	88	55	88	57
VotedP	84	88	84	53	84	54
NaiveB	81	86	81	49	81	50
ZeroR	77	82	77	47	77	48

Table 4: **Performance % (Intuitive Task):** all training data used, post–evaluation studies

	Precision		Recall		F-Score	
Method	mic	mac	mic	mac	mic	mac
Mean	**91**	**78**	**90**	**60**	**90**	**60**
J48-AB	92	61	92	60	92	60
JRip-AB	92	61	92	60	92	60
JRip	92	61	92	60	92	60
J48	90	60	90	59	90	59
SMO	88	92	88	55	88	57
NaiveB	81	86	81	49	81	50
ZeroR	77	82	77	47	77	48

Table 5: **Performance % (Textual Task):** training data limited to dominant judgments, results of our official submissions plus post–evaluation studies

	Precision		Recall		F-Score	
Method	mic	mac	mic	mac	mic	mac
Mean	**91**	**75**	**91**	**56**	**91**	**56**
JRip-AB	*93*	*96*	*93*	*45*	*93*	*46*
JRip	*93*	*96*	*93*	*45*	*93*	*46*
J48-AB	*93*	*96*	*93*	*45*	*93*	*46*
J48	91	95	85	42	88	44
SMO	88	94	88	41	88	42
VotedP	85	91	85	40	85	41
NaiveB	81	89	81	36	81	37
ZeroR	78	87	78	34	78	35

Table 6: **Performance % (Textual Task):** all training data used, post–evaluation studies

	Precision		Recall		F-Score	
Method	mic	mac	mic	mac	mic	mac
Mean	**91**	**75**	**91**	**56**	**91**	**56**
J48-AB	93	48	92	45	93	46
J48	92	48	91	45	92	46
JRip	93	46	93	46	93	46
JRip-AB	93	71	93	46	92	46
SMO	88	94	88	41	88	42
NaiveB	81	89	81	36	81	37
ZeroR	78	87	78	34	78	35

both macro and micro definitions. The macro definition computes the score per judgment, and then averages those scores, while the micro definition computes overall scores. Thus, the macro definition will reward systems that fare well on the minority judgments, and the micro definition will do the same for systems that perform well on the dominant judgments. The overall ranking of systems in the Challenge was done according to the Macro F-Score.

In Tables 3, 4, 5, and 6, the first entry is the mean value of the 28 submitted systems for each of the measures. These scores (and team ranks) were provided by the Challenge organizers after the evaluation period was over. The last entry in each table is the results from the ZeroR classifier. This is a majority classifier that assigns the most frequent judgement found in the training data for a morbidity to each discharge summary in the test data. A majority classifier provides a lower bound for the performance that a supervised learning algorithm should obtain.

Table 3 shows the official results of our three submitted systems on the intuitive task (in italics), which are designed as J48-AB, JRip-AB, and JRip. An -AB following the name of a learning algorithm indicates that AdaBoost was applied. For the intuitive task our Macro F-scores for all three systems are slightly above the mean of the 28 participating systems, and our Precision scores are quite a bit higher. The rank of our system with the highest F-Macro score (JRip) in the intuitive task was 17 of 28.

After the evaluation period was over and once the annotations for the test data were available, we decided to re–run our systems, except this time using all of the available training data, just to see if there would be any difference in the results. We show the results of this post–evaluation study in Table 4. It turned out that our results would be essentially the same in the intuitive task with or without the inclusion of the minority training examples.

Table 5 shows the results of our submitted systems on the textual task. Here we can see that our results trailed the mean results by a significant margin. Our best system according to the Macro F-score (JRip-AB) ranked 21 out of 28 participating systems. Our Macro Recall lagged below the mean, while all of the other Precision and Recall measures were somewhat above the mean

performance.

While the number of minority judgments was still quite small in the textual task, it was somewhat more than in the intuitive task. We were again uncertain if excluding the minority judgments from the training data might have needlessly hurt our results. We did the same post–evaluation study as we did in the intuitive task, where we re–ran our textual systems using all of the training data and evaluated their performance on the test data. Table 6 shows the results, where we find that again there is not a significant difference in the performance compared to training with just the dominant judgments.

In retrospect, the fact that the minority judgments had no impact on supervised learning even when included in the training data is not too surprising, since supervised learning algorithms strive to learn a model that covers or explains the training data as completely as possible, without over–training. In order to make the models generalize, there is usually a pruning component of some kind after the model has been learned, and in this case the minority judgments occur so rarely (just a few times for a given morbidity at the most) that they are indistinguishable from noise, and so the supervised learning algorithm simply discards them.

Example of JRip Rules

The following rule set was learned by JRip for Depression in the intuitive task. While this might appear very similar to the information in a decision tree, these rules were constructed in a bottom–up fashion rather than the top–down method of decision tree learning.

```
(depression = 1) AND
(catheterization = 0) => Y (87.0/8.0)
(depression = 1) AND
(exercise = 0) => Y (22.0/7.0)
(wellbutrin = 1) => Y (4.0/0.0)
(citalopram = 1) => Y (2.0/0.0)
(celexa = 1) => Y (14.0/0.0)
(zoloft = 1) => Y (11.0/1.0)
(prozac = 1) => Y (7.0/0.0)
(paxil = 1) => Y (7.0/1.0)
ELSE => N (543.0/5.0)
```

These rules have an intuitive interpretation. For example, the first rules states: *If the string depression occurs in the discharge summary and catheterization does not, then Depression is present (Y).* Each rule is considered in turn, and if none of the Y conditions are met, then the output will be N, indicating that depression is not present.

The numbers in parenthesis indicate how many of the training examples were covered successfully by this rule. For example, for the first rule, 87 examples were covered by this rule correctly, and 8 were not (they met the conditions but had a judgment of N). This set of rules resulted in a Macro F-Score of 91%.

Conclusion

Our approach to the I2B2 Obesity Challenge was to treat this as a problem in supervised learning, where we learned a classifier for each morbidity based on the unigrams that appear in the discharge summaries. These classifiers were learned with Ripper/JRip and C4.5/J48, where JRip was run with and without AdaBoost, and J48 was run with AdaBoost. We ignored the very small number of minority judgments present in each task and did not include those examples in the training data. Subsequent analysis after the release of the test data showed that our results would have been approximately the same whether we included the minority judgments or not.

Acknowledgments

The author would like to thank the I2B2 Challenge Organizers for their efforts in creating the training and evaluation data, and for the very professional manner in which the Challenge was conducted.

Address for Correspondence

Ted Pedersen, University of Minnesota, Department of Computer Science 1114 Kirby Drive, Duluth, MN 55812-2496, USA
tpederse@d.umn.edu

References

1. S. Banerjee and T. Pedersen. The design, implementation, and use of the Ngram Statistics Package. In *Proceedings of the Fourth International Conference on Intelligent Text Processing and Computational Linguistics*, pages 370–381, Mexico City, February 2003.
2. I. Witten and E. Frank. *Data Mining - Practical Machine Learning Tools and Techniques with Java Implementations*. Morgan–Kaufmann, San Francisco, CA, second edition, 2005.
3. W. Cohen. Fast effective rule induction. In *Proceedings of Twelfth International Conference on Machine Learning*, pages 115–123, Tahoe City, CA, July 1995.
4. J. Quinlan. Induction of decision trees. *Machine Learning*, 1:81–106, 1986.
5. Y. Freund and R.E. Schapire. Experiments with a new boosting algorithm. In *Proceedings of Thirteenth International Conference on Machine Learning*, pages 148–156, Bari, Italy, 1996.