Boosting Trees for Anti-Spam Email Filtering

Xavier Carreras and Lluís Màrquez

TALP Research Center
LSI Department
Technical University of Catalonia (UPC)
{carreras,lluism}@lsi.upc.es

5th September 2001

RANLP'01

Outline

- Introduction
- AdaBoost Learning Algorithm
- Spam-Filtering Corpora
- Evaluation Measures
- Experiments and Results
- Conclusions and Future Research

Introduction

Anti-Spam Email Filtering is the problem of automatically identifying unwanted electronic mail messages.

- It is a particular problem of Text Categorization:
  - Two classes: *spam* or *legitimate*.
  - A message is *spam* for its intention, rather than its subject.
  - Particular forms, vocabulary and word patterns.

- A *legitimate* message must not be filtered!
  High-precision filters are desired.

- Many automatic IR techniques have been applied:
  - Naive Bayes (Sahami et al., 98)
  - SVM's, Boosting, RIPPER, & Rocchio (Drucker et al., 99)
  - NB vs. TiMBL (Androutsopoulos et al. 00)
  - Stacking of Classifiers (Sakkis et al. 01)

Our Approach

We apply AdaBoost to the anti-spam problem.

We obtain:

- Very good performance on the evaluation corpora.
- High-precision classifiers, exploiting the confidence of predictions.

AdaBoost

- General method for combining classifiers, called *weak classifiers*.
- Schapire & Singer, 1999
- Success on several NLP problems:
  - Text Categorization (Schapire & Singer, 00)
  - Word Sense Disambiguation (Escudero et al., 00)
  - PP-attachment (Abney et al., 99)
  - POS-tagging (Abney et al., 99)
  - Syntactic Clause Splitting (Carreras & Màrquez, 01)
- Some properties:
  - Designed theoretically.
  - Easy to implement.
  - Resistant to overfitting.
  - Able to manage very large feature spaces.
  - Predictions are given with a *confidence score*.

AdaBoost Procedure

procedure AdaBoost (**in** $X = \{(x_i, y_i)\}_{i=1}^{m}$)

$D_1(i) = 1/m$
for $t = 1$ **to** T **do**

$h_t = WeakLearner(X, D_t)$
Choose $\alpha_t \in \mathbb{R}$
Update the distribution:

$$D_{t+1}(i) = \frac{D_t(i)\ e^{(-y\alpha_t h_t(x))}}{Z_t}$$

...with Z_t being a normalization factor.

end for
Output the classifier:

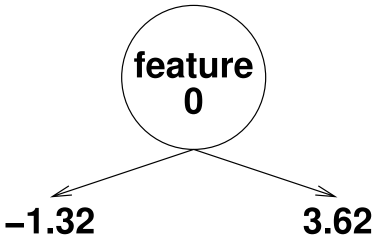
$$f(x) = \sum_{t=1}^{T} \alpha_t h_t(x)$$

end procedure

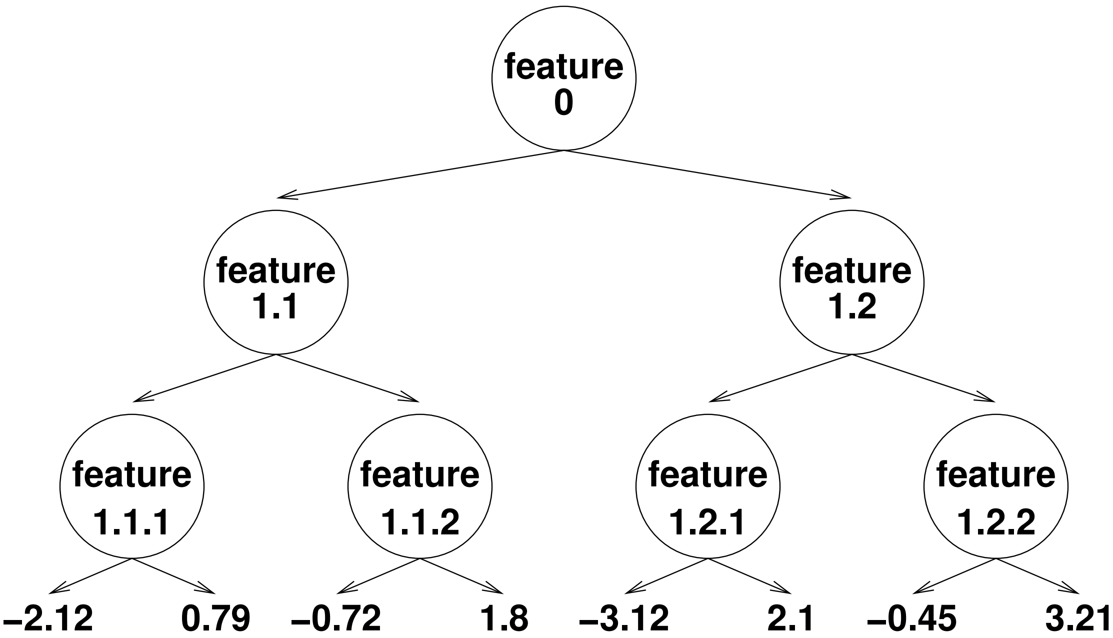
Weak Rules

- Domain-partitioning weak rules.
  Designed for AdaBoost in (Schapire & Singer, 99).

- Stumps: AdaBoost of one-node trees.



- TreeBoost[n]: AdaBoost of trees of fixed depth n.



AdaBoost Decision Scheme

Given an AdaBoost classifier

$$f(x) = \sum_{t=1}^{T} \alpha_t h_t(x) \qquad f(x) \in \mathbb{R}$$

- Decision threshold: θ
- An example x is classified positive iff $f(x) \geq \theta$.
- $\left|f(x)\right|$ is the *confidence* of the prediction.
- Default setting: $\theta = 0$.
- Increasing θ results in a more precise filter.

Evaluation Corpora

- Two public Corpora for Anti-Spam filtering (Androutsopoulos et al, 00):
  - **PU1:** personal messages vs. spam
  - **LingSpam:** Linguist list messages vs. spam

	PU1	Lingspam
Messages	1,099	2,893
Spam	481	481
Legitimate	618	2,412
Spam %	43.7%	16.6%
Vocabulary	26,449	65,723

- Features: bag-of-words model.
  - No stemming.
  - No stop-word removal.

- 10 fold cross-validation.

Evaluation Measures

Accuracy The percentage of correct predictions.

Precision The number of correctly classified spams, out of the number of messages predicted as spam.

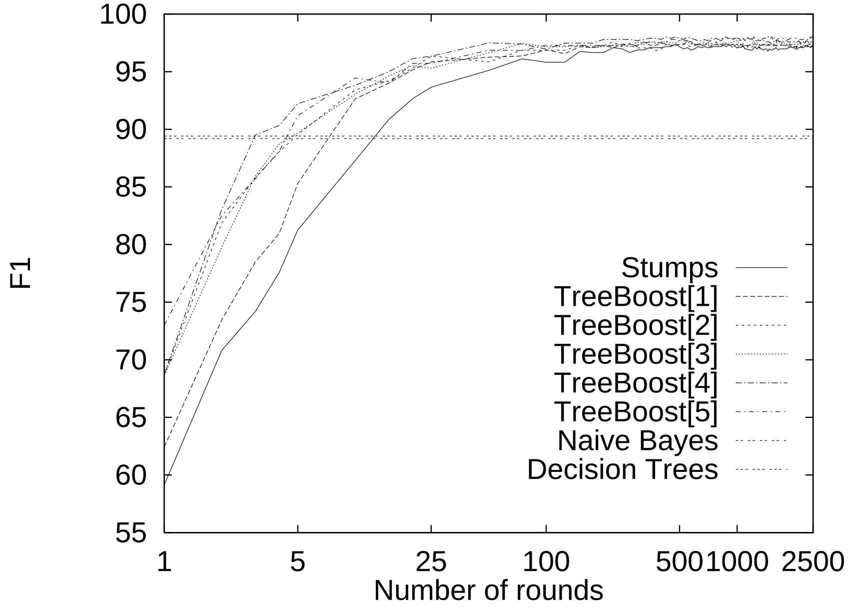
Recall The number of correctly predicted spams, out of the number of spams in the corpus.

F_β Interpolation between precision and recall:

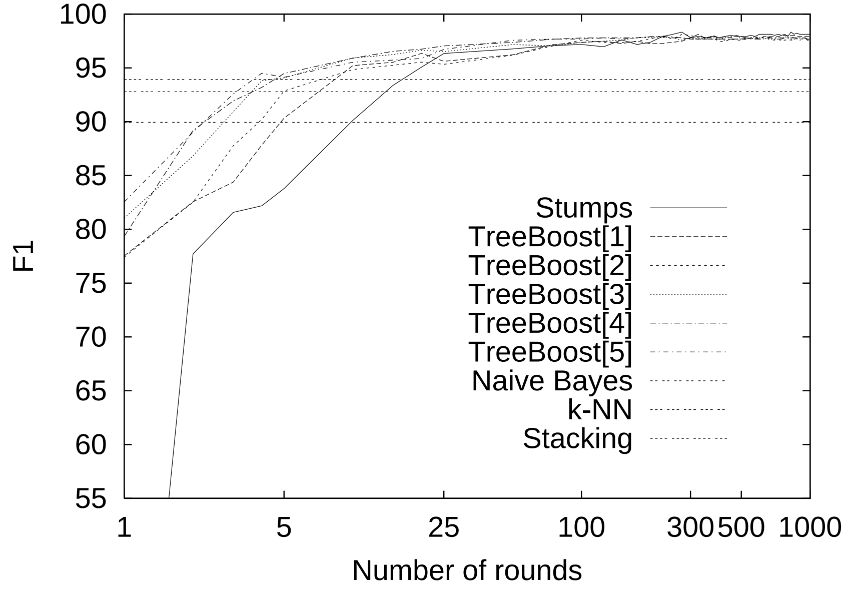
$$F_1 = \frac{2 \cdot precision \cdot recall}{precision + recall}$$

Comparing Methods

PU1



Lingspam



Comparing Methods

PU1

	T	recall	precision	F_1
N. Bayes		83.98	95.11	89.19
D. Trees		89.81	88.71	89.25
Stumps	525	96.47	97.48	96.97
TreeBoost[3]	675	96.88	97.90	97.39
TreeBoost[5]	550	96.88	98.52	97.39

LingSpam

	T	recall	precision	F_1
N. Bayes		82.40	99.00	89.94
k-NN		88.60	97.40	92.79
Stacking		91.70	96.50	93.93
Stumps	1,000	97.92	98.33	98.12
TreeBoost[3]	1,000	96.47	98.93	97.68
TreeBoost[5]	1,000	96.26	99.14	97.68

High-Precision Classifiers

- **Goal:** Obtain classifiers which do not filter *legitimate* messages.

- The $\lambda - cost$ defines the strictness of a scenario:

  "Misclassifying a legitimate message is like misclassifying λ spams."

- Two scenarios are proposed in (Androutsopoulos, 00):

$\lambda = 9$	supervised filter
$\lambda = 999$	automatic filter

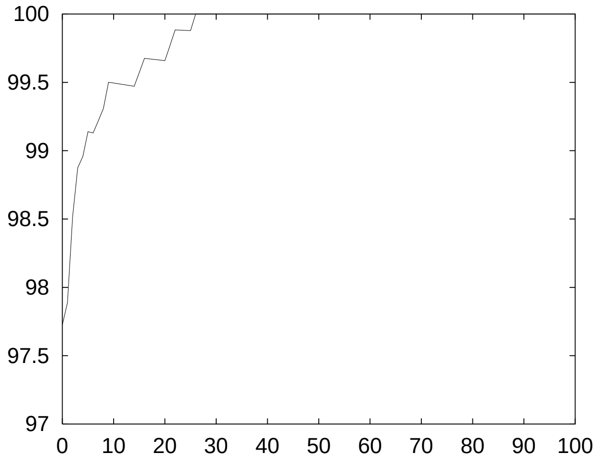
- The *weighted accuracy* evaluates the behaviour of a classifier in a particular scenario:

$$WAcc = \frac{\lambda \cdot L_{-} + S_{+}}{\lambda \cdot L + S}$$

- We exploit the *confidence* of the predictions for tuning the classifiers.

Rejection Curves

The accuracy without the $n\%$ of the less confident predictions.



Rejection Curve for TreeBoost[3]

It can be stated that ...

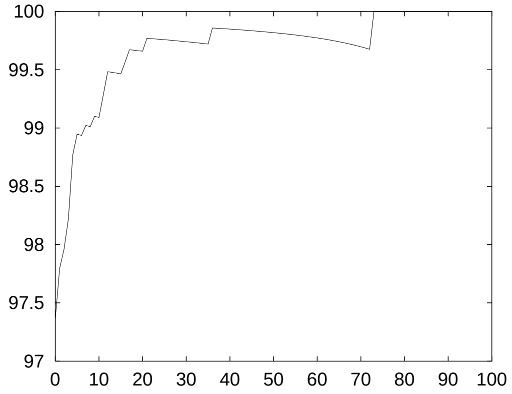
- Misclasifications are produced only on the less confident predictions.

- As the deep of the weak rules increases, the predictions slightly improve.

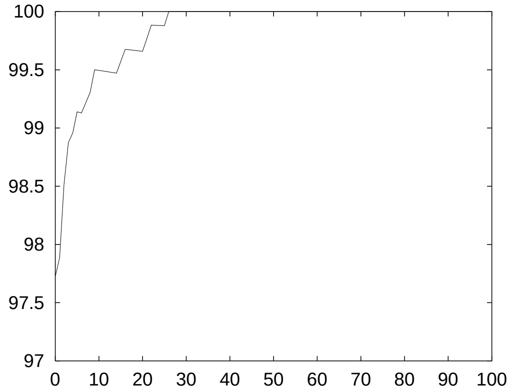
Rejection Curves

PU1

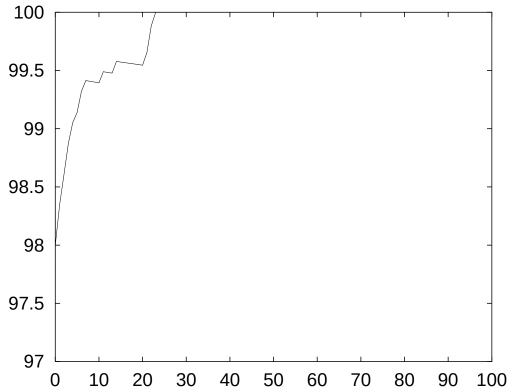
Stumps



TreeBoost[3]

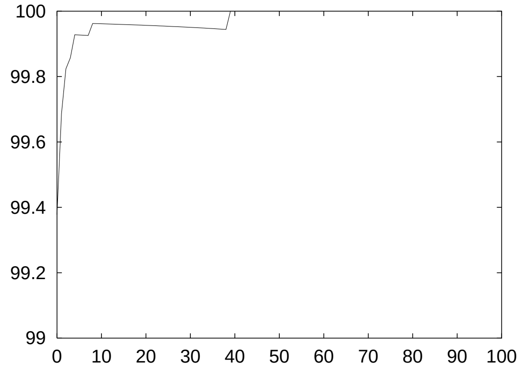


TreeBoost[5]

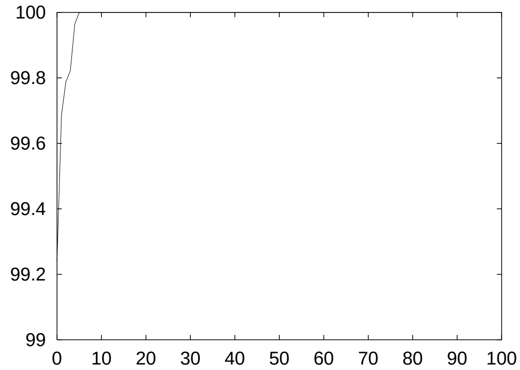


LingSpam

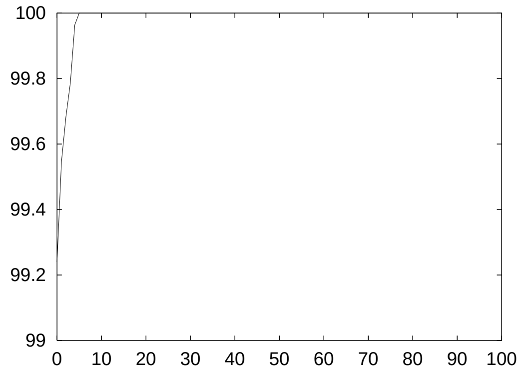
Stumps



TreeBoost[3]



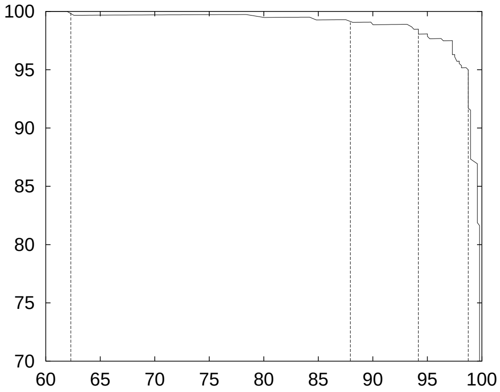
TreeBoost[5]



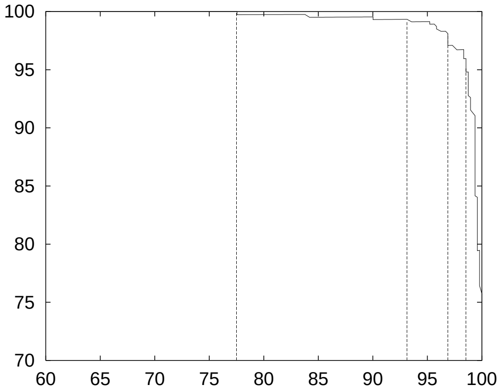
Precision-Recall Curves

PU1

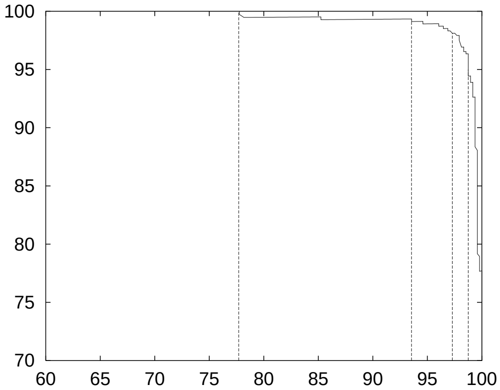
Stumps



TreeBoost[3]

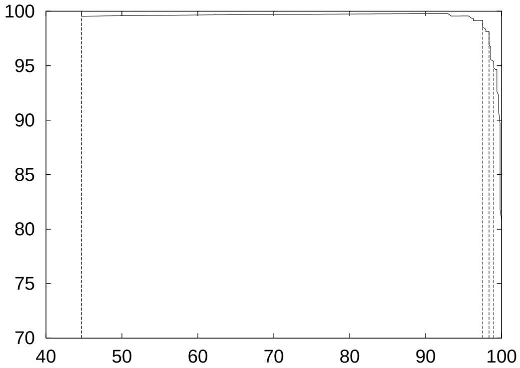


TreeBoost[5]

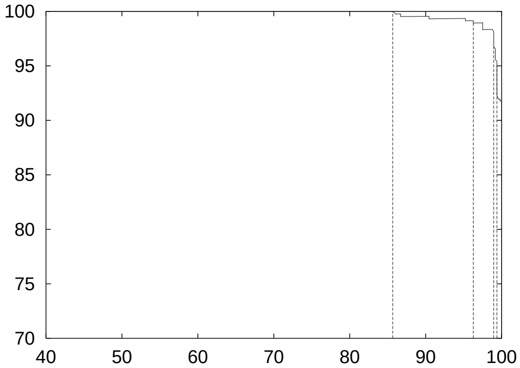


LingSpam

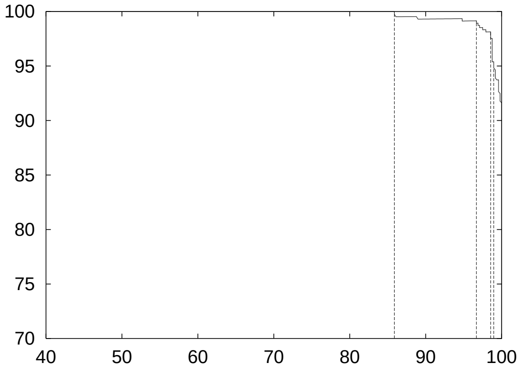
Stumps



TreeBoost[3]



TreeBoost[5]



High Precision Classifiers

- The decision threshold θ can be tuned to obtain a high precision classifier.
- A validation set is used for tuning θ.
  Given 9 training folds in the CV process:
  - A classifier is learned using 8 folds.
  - The remaining fold forms the validation set.
  - The value θ which optimizes $WAcc_{\lambda}$ on the validation fold is selected.

Cost-Sensitive Results on the PU1

$\lambda = 9$	θ	Recall	Prec.	WAcc
Naive Bayes	-	78.77	96.65	96.38
Stumps	9.16	89.60	99.08	98.58
TreeBoost[1]	10.20	93.55	98.71	98.59
TreeBoost[2]	24.15	93.76	98.90	98.76
TreeBoost[3]	45.16	91.48	99.32	98.87
TreeBoost[4]	19.06	94.80	99.35	99.14
TreeBoost[5]	37.39	93.97	99.12	98.92

$\lambda = 999$	θ	Recall	Prec.	WAcc
Naive Bayes	-	46.96	98.80	99.47
Stumps	16.29	84.20	99.51	99.66
TreeBoost[1]	46.87	74.43	100	99.98
TreeBoost[2]	96.71	76.30	100	99.98
TreeBoost[3]	126.38	74.01	100	99.98
TreeBoost[4]	123.09	64.45	100	99.97
TreeBoost[5]	177.909	66.53	100	99.97

Cost-Sensitive Results on the LingSpam

$\lambda = 9$	θ	Recall	Prec.	WAcc
Naive Bayes	-	77.57	99.45	99.43
TiMBL	-	74.05	98.97	99.27
k-NN	-	88.6	97.4	99.40
Stacking	-	84.8	98.8	99.46
Stumps	7.028	96.47	99.15	99.76
TreeBoost[1]	25.949	94.80	99.35	99.77
TreeBoost[2]	31.178	95.22	98.92	99.69
TreeBoost[3]	32.492	95.84	99.14	99.75
TreeBoost[4]	26.022	95.84	99.14	99.75
TreeBoost[5]	-9.107	96.26	99.14	99.76

$\lambda = 999$	θ	Recall	Prec.	WAcc
Stumps	81.91	52.81	99.61	99.95
TreeBoost[1]	217.12	50.94	100	99.99
TreeBoost[2]	209.51	77.55	99.73	99.95
TreeBoost[3]	342.58	74.22	100	99.99
TreeBoost[4]	337.19	81.91	100	99.99
TreeBoost[5]	451.13	79.83	100	99.99

Conclusions

- AdaBoost outperforms other methods on the evaluated spam corpora: PU1 and LingSpam. High F_1 rates above 97% are achieved.
- High-precision classifiers are obtained using the confidence of the predictions:
  - Missclassifications occur only on the less confident predictions.
  - The decision threshold can be tuned to obtain a desired behaviour.
  - The depth of AdaBoost Weak Rules improves the confidence of the predictions.
  - The performance seems good enough to construct a real system.

Future Research

- Study the portability of the presented experimental methods to more difficult corpora or other filtering domains.

- Introduce the misclassifications costs into the learning algorithm.