Naïve Bayes, Text classification and sentiment

Imran Khan

Important: This lecture slides borrow heavily from the third edition of Stanford book “Speech and Language Processing” authored by Daniel Jurafsky and James Martin

Text classification

- Naïve Bayes is a supervised learning algorithm used for text classification. And text classification is the task of assigning an entire text document one of labels (or one of categories).

- **Common text categorization task:**

a. **Sentiment analysis:** Assigning a positive or negative sentiment to the opinion of writers towards some object.

b. **Spam detection:** Classifying an email as spam versus non-spam.

c. **Authorship attribution:** determining the authorship of text.

d. **Language identification:** Often the first step is to identify the language. Because the text on social media is written on many languages.

e. **Topic labelling :** Assigning a topic to an entire text document.

Generative versus Discriminative

Discriminative model:

In discriminative model, we directly compute probability of class c given document d, $P(c|d)$

How does the discriminative model learn?
Model puts higher weight on features that directly improves its ability to discriminate between possible classes.

Generative model

- How does naïve Bayes build the ML model? It assigns a class **c** to document **d** *not* by directly computing $P(c|d)$ but by computing **likelihood** and **prior.**

$$\hat{c} = \underset{c \in C}{\operatorname{argmax}} \overbrace{P(d|c)}^{\text{likelihood}} \overbrace{P(c)}^{\text{prior}}$$

Generative model
How to generate features of a document if we know if it was of class c?

Naïve Bayes is a probabilistic classifier

Naive Bayes is a probabilistic classifier, meaning that for a document d, out of all classes c ∈ C the classifier returns the class ˆc which has the maximum posterior ˆ probability given the document.

$$\hat{c} = \underset{c \in C}{\operatorname{argmax}} P(c|d)$$

How to read this argmax?

$$P(x|y) = \frac{P(y|x)P(x)}{P(y)}$$

Bayes theorem
Posterior probability?
Likelihood?
Prior?

$$\hat{c} = \underset{c \in C}{\operatorname{argmax}} P(c|d) = \underset{c \in C}{\operatorname{argmax}} \frac{P(d|c)P(c)}{P(d)}$$

We can drop the denominator P(d). Do you know why?

Whether the email is spam or non-spam depends on which class has maximum posterior probability?

Naïve Bayes

$$\hat{c} = \underset{c \in C}{\operatorname{argmax}} P(c|d) = \underset{c \in C}{\operatorname{argmax}} P(d|c)P(c)$$

What class label will be assigned to a document? The class label that will maximize the product of likelihood and prior.

$$\hat{c} = \underset{c \in C}{\operatorname{argmax}} \overbrace{P(f_1, f_2, \ldots, f_n|c)}^{\text{likelihood}} \overbrace{P(c)}^{\text{prior}}$$

We can represent a document d as a set of features.

It is still difficult to compute likelihood without some simplifying assumptions. Let us first try to understand why is it difficult to compute likelihood? Why do we need impossibly large dataset to compute every possible combination of features?

Two simplifying assumptions

1. Conditional independence assumption
2. Bag of words assumption

$$P(f_1, f_2, ..., f_n|c) = P(f_1|c) \cdot P(f_2|c) \cdot ... \cdot P(f_n|c)$$

With two simplifying assumptions we transform the likelihood on the last slide to the simple multiplication of individual features given the class.

$$c_{NB} = \underset{c \in C}{\operatorname{argmax}} P(c) \prod_{f \in F} P(f|c)$$

Conditional independence assumption

- Let us take an example of spam detection, certain words co-occur in spam emails.

- Congratulations! You win a million-dollar lottery.

It is a spam email

Given you have seen a word “win” in a spam email, does the use of word “lottery” in the same spam email is

A. more probable
B. less probable
C. Equally probable

Conditional independence assumption is a naïve (stupid) assumption, but it still produces surprisingly good result.

Bag of words model

- Naïve Bayes and Logistic regression is a bag of words model. But what is bag of words model?
- In bag of words model, “We represent a text document as if it were a bag of words, that is, an unordered set of words with their position ignored, keeping only their frequency in the document.”
- The bag of words assumption means that the position of words do not matter for understanding the text document and “that the word “love” has the same effect on classification whether it occurs at the 1st, 20th, or last word in the document.”
- Obviously, bag of words is a very strong assumption for any human language. The semantic and syntactic relationship of the text depends on the order of the words in which they appear.
- Interesting example: **"Dog bites man."** vs. **"Man bites dog."**

Bag of words model examples

- Interesting example:

- **"Dog bites man."** vs. **"Man bites dog."**

Same three words in different order have different meaning

- **"The cat chased the mouse."** vs. **"The mouse chased the cat."**

This sentence reflects the usual predator-prey relationship

Use of same words in different order. This sentence reflects the very unusual situation.

These examples conveys the essential idea that **overall meaning** of the sentence depends on the order of the words in which they appear

Training the naïve Bayes classifier

$$c_{NB} = \underset{c \in C}{\operatorname{argmax}} P(c) \prod_{f \in F} P(f|c)$$

$$c_{NB} = \underset{c \in C}{\operatorname{argmax}} P(c) \prod_{i \in positions} P(w_i|c)$$

Features are nothing but words used in a document.

$$\hat{P}(c) = \frac{N_c}{N_{doc}}$$

$$\hat{P}(w_i|c) = \frac{count(w_i, c)}{\sum_{w \in V} count(w, c)}$$

This is computed based on computing simple frequency from training dataset.
What fraction of the total documents belong to class c?

There is one more problem!

- Imagine we are trying to estimate the probability of the word “fantastic” given the class “positive” but suppose there are no training documents that both contain the word “fantastic” are classified as positive.

$$\hat{P}(\text{“fantastic”}|\text{positive}) = \frac{count(\text{“fantastic”}, \text{positive})}{\sum_{w \in V} count(w, \text{positive})} = 0$$

Since the naïve bayes naively multiplies all the individual probabilities of words in a likelihood term, therefore, zero probability of any one word given a class in a likelihood term will reduce the whole likelihood term to zero.

Laplace smoothing handles the issue of a zero probability!

- Add one Laplace smoothing address this issue by adding the count of one in each word(feature). In other words, we assume that each word appears at least once in each class.

$$\hat{P}(w_i|c) = \frac{count(w_i,c)+1}{\sum_{w\in V}(count(w,c)+1)} = \frac{count(w_i,c)+1}{\left(\sum_{w\in V} count(w,c)\right)+|V|}$$

Worked example!

	Cat	Documents
Training	-	just plain boring
	-	entirely predictable and lacks energy
	-	no surprises and very few laughs
	+	very powerful
	+	the most fun film of the summer
Test	?	predictable with no fun

Sentiment classification problem

We are trying to classify the test sentence either as positive or negative. Therefore, we have to compute posterior probabilities for both classes. And what class label we will assign to this test sentence depends on higher posterior class probability

1. Probability (+ |test sentence)
2. Probability(-|test sentence)

$$P(-) = \frac{3}{5} \qquad P(+) = \frac{2}{5}$$

$$P(\text{“predictable”}|-) = \frac{1+1}{14+20} \qquad P(\text{“predictable”}|+) = \frac{0+1}{9+20}$$

$$P(\text{“no”}|-) = \frac{1+1}{14+20} \qquad P(\text{“no”}|+) = \frac{0+1}{9+20}$$

$$P(\text{“fun”}|-) = \frac{0+1}{14+20} \qquad P(\text{“fun”}|+) = \frac{1+1}{9+20}$$

Worked example!

$$P(-)P(S|-) = \frac{3}{5} \times \frac{2 \times 2 \times 1}{34^3} = 6.1 \times 10^{-5}$$

$$P(+)P(S|+) = \frac{2}{5} \times \frac{1 \times 1 \times 2}{29^3} = 3.2 \times 10^{-5}$$

Evaluation: Precision, recall, F-measure

- **Confusion matrix for binary class classification problem**

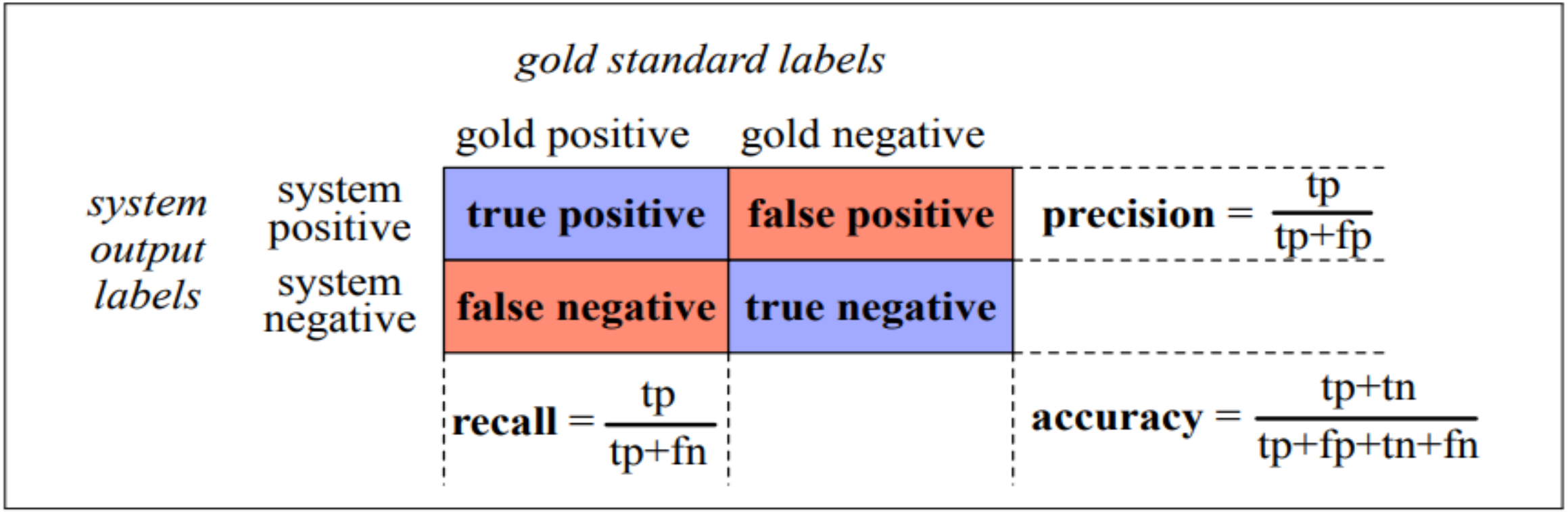

Figure 4.4 A confusion matrix for visualizing how well a binary classification system performs against gold standard labels.

Is accuracy a good measure to evaluate the performance of classification model?
Or when accuracy is not a good measure to evaluate the performance of the classification model?

Confusion matrix jargon

- **True Positive:** The model correctly classify the actual spam email as spam.
- **False Positive:** The model incorrectly classify the actual non-spam email as spam email.
- **True Negative:** The model correctly classify the non-spam email as non-spam.
- **False Negative:** The model incorrectly classify the actual spam email as non-spam email.

Do you think false negative and false positive are equally bad?

Confusion matrix jargon

- **Precision:** In case of precision, we are trying to answer, the fraction of emails classified by the model as spam emails were actually spam emails. What do we mean by low precision? By low precision, we mean that the good number of non-spam (legitimate) emails will be classified as spam emails. Therefore, we prefer high precision than low precision.

- **Recall:** In case of recall, we are trying to answer, what fraction of actual spam emails were classified as spam. What do we mean by low recall? By low recall, we mean that the good number of spam emails are classified by the model as non-spam. Therefore, we prefer high recall than low recall.

We don’t want spam email to end up in our inbox folder. And at the same time we don’t want non-spam email (legitimate) emails to end up in spam folder. Therefore, we want “high recall” and high precision at the same time.

Why is accuracy not a good measure if the classes are unbalanced?

- Let us suppose, out of one million emails, 999,900 emails are non-spam and only 100 emails are spam. So, if the model naively classify every email as non-spam, then the model will have the accuracy of 99.99. But would that model (classifier) with so high accuracy be any useful? It did not correctly classify any of spam email as spam. Instead, it will classify spam email as non-spam. We are interested in predicted the rare class (spam email, prevalence of heart disease etc.) correctly?

		Actual Labels	
		Spam email	**Non-spam email**
Predicted Labels	**Spam email**	0	0
	Non-spam email	100	999,900

What is a recall?
Precision is equally problematic.

False positive and False negative are not equally bad

Spam Detection (Email Filtering):

False Positives (FP): Legitimate emails marked as spam.

Impact: Important emails may be missed, leading to potential loss of information or communication breakdown.

False Negatives (FN): Spam emails not detected and reaching the inbox.

Impact: Users receive spam, which can be annoying and may contain harmful content like phishing or malware.

Medical Diagnosis:

False Positives: Healthy individuals incorrectly diagnosed with a disease.

Impact: Unnecessary stress, further invasive testing, and possibly harmful treatments.

False Negatives: Diseased individuals incorrectly diagnosed as healthy.

Impact: Missed treatment opportunities, progression of the disease, and potentially fatal outcomes.

Fraud Detection:

False Positives: Legitimate transactions flagged as fraudulent.

Impact: Customer inconvenience, potential loss of business, and administrative burden.

False Negatives: Fraudulent transactions not detected.

Impact: Financial loss, damage to reputation, and potential legal issues.

F1-score

- F1-score is the single metric that incorporates both precision and recall simultaneously. Higher F1 score is better.

$$\text{F1 Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

F1- score is important metric specially when the class imbalance is severe.

Evaluating with more than two classes

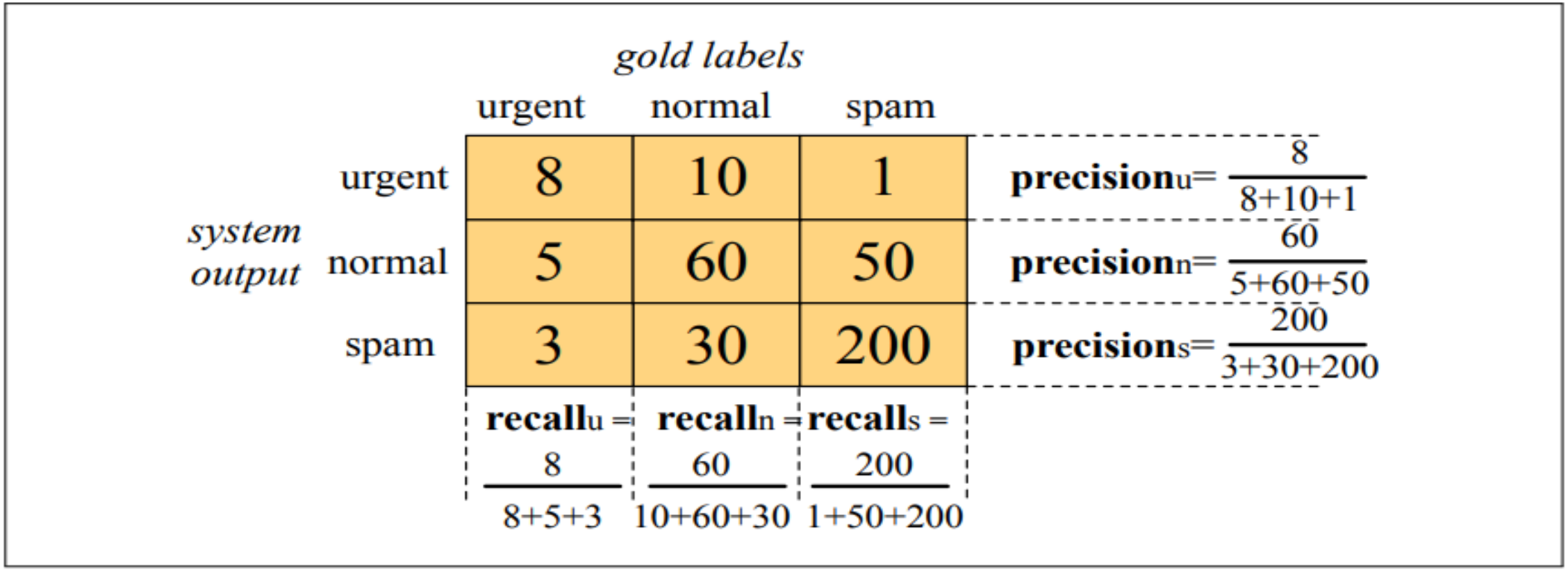

Figure 4.5 Confusion matrix for a three-class categorization task, showing for each pair of classes (c_1, c_2), how many documents from c_1 were (in)correctly assigned to c_2.

Here we have computed distinct precision and recall for each class.

Single metric

- How to derive a single measure that tells us how well the classifier (model/system) is doing the classification task? We have two approaches to derive a single measure. 1. macro averaging 2. micro averaging

- In **Macroaveraging,** we compute the performance for each class, and then average over classes.

- In **microaveraging,** we collect the decisions for all classes into a single confusion matrix, and then compute precision and recall from that table.

Microaveraging versus Macroaveraging

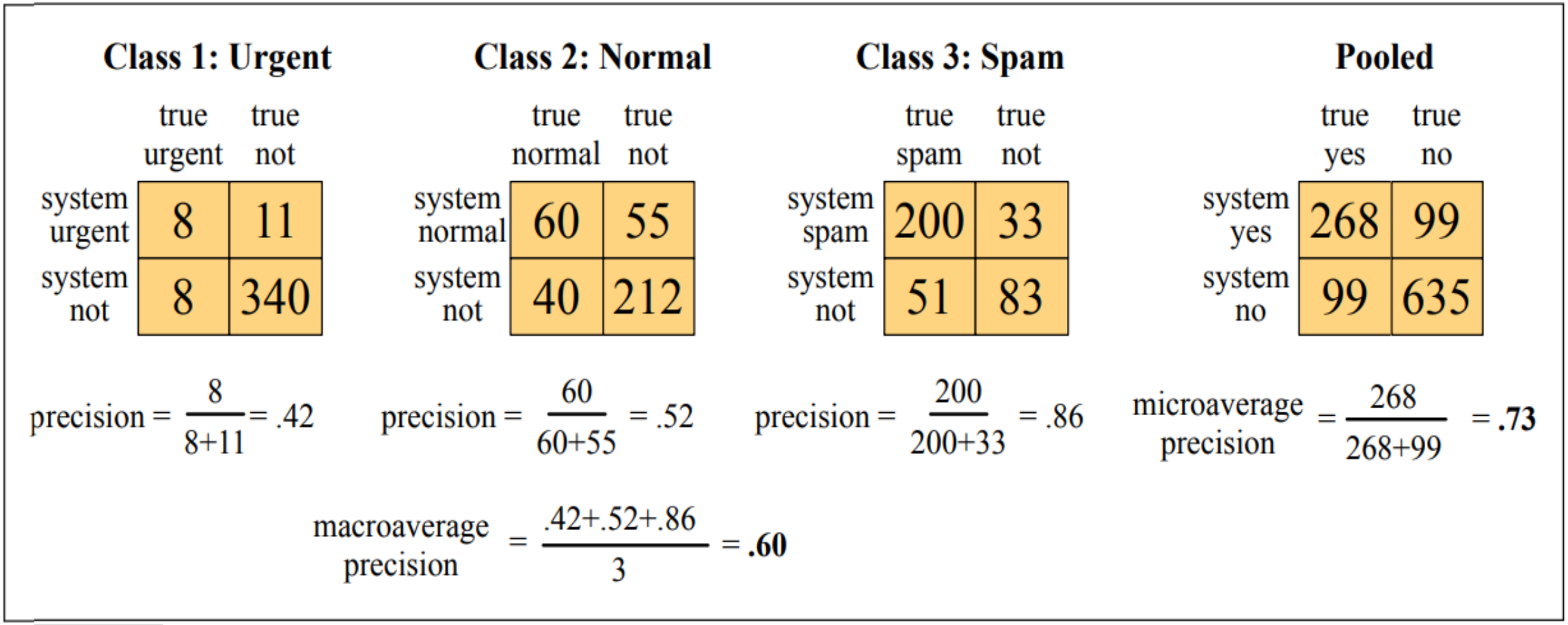

Figure 4.6 Separate confusion matrices for the 3 classes from the previous figure, showing the pooled confusion matrix and the microaveraged and macroaveraged precision.