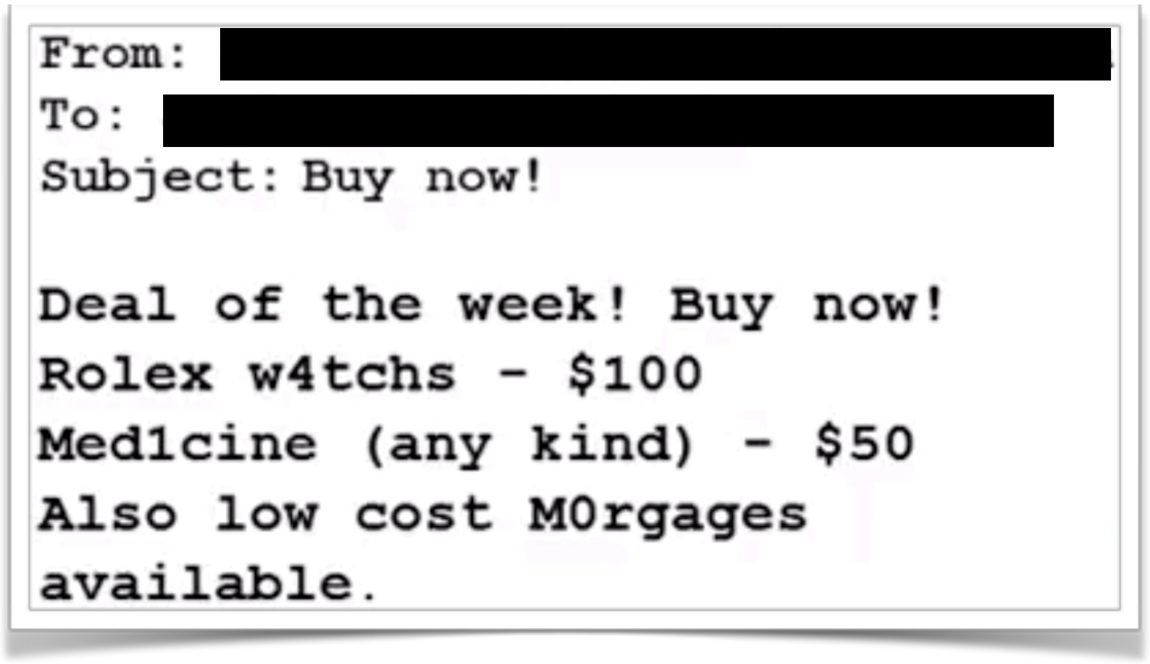
Machine Learning System Design

Teeradaj Racharak (เอ็กซ์)
r.teeradaj@gmail.com

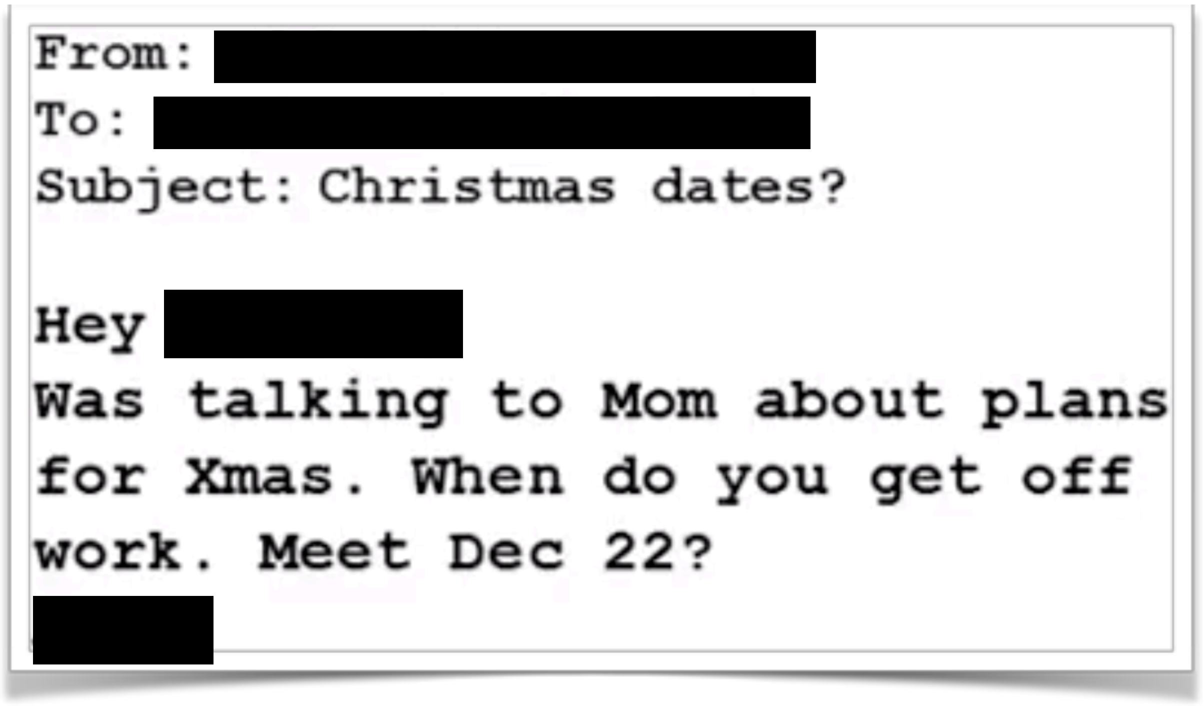
Spam Classification Example

Motivating Example



```
From: [redacted]
To: [redacted]
Subject: Buy now!

Deal of the week! Buy now!
Rolex w4tchs - $100
Med1cine (any kind) - $50
Also low cost M0rgages
available.
```



```
From: [redacted]
To: [redacted]
Subject: Christmas dates?

Hey [redacted]
Was talking to Mom about plans
for Xmas. When do you get off
work. Meet Dec 22?
[redacted]
```

Spammed Email
(Labeled with 1)

Non-spammed Email
(Labeled with 0)

Building a Spam Classifier

Supervised learning *e.g.* logistic regression, (multinomial) naive Bayes

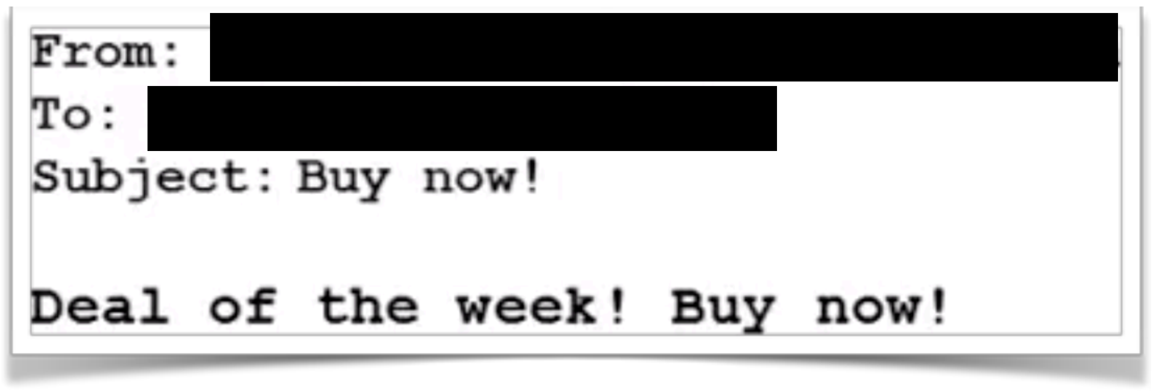
- x = features of email
- y = spam (1) or not spam (0)

Features x : Choose 100 words indicative of spam / not spam

e.g. deal, buy, discount, teeradaj, now

$$x = \left[\begin{array}{c} 1 \\ 1 \\ 0 \\ 1 \\ \vdots \\ 0 \\ \vdots \end{array}\right] \begin{array}{l} \text{buy} \\ \text{deal} \\ \text{discount} \\ \text{now} \\ \quad\vdots \\ \text{teeradaj} \\ \quad\vdots \end{array}$$

one hot


```
From: [redacted]
To: [redacted]
Subject: Buy now!

Deal of the week! Buy now!
```

$$x_j = \begin{cases} 1 & \text{if } word_j \text{ appears in email} \\ 0 & \text{otherwise} \end{cases}$$

Building a Spam Classifier

Supervised learning *e.g.* logistic regression, (multinomial) naive Bayes

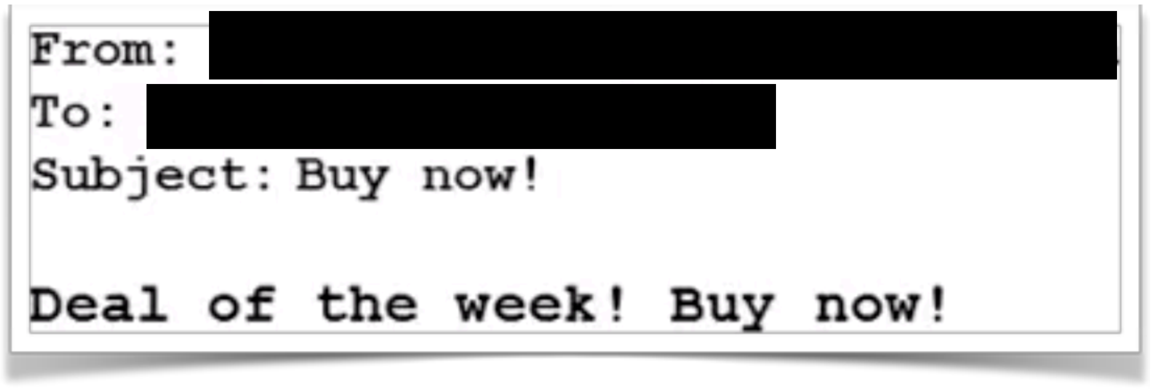
- x = features of email
- y = spam (1) or not spam (0)

Features x : Choose 100 words indicative of spam / not spam

e.g. deal, buy, discount, teeradaj, now

$$x = \begin{array}{c l} \begin{bmatrix} 1 \\ 1 \\ 0 \\ 1 \\ \vdots \\ 0 \\ \vdots \end{bmatrix} & \begin{array}{l} \text{buy} \\ \text{deal} \\ \text{discount} \\ \text{now} \\ \vdots \\ \text{teeradaj} \\ \vdots \end{array} \end{array}$$

one hot



```
From: [redacted]
To: [redacted]
Subject: Buy now!

Deal of the week! Buy now!
```

$$x_j = \begin{cases} 1 & \text{if } word_j \text{ appears in email} \\ 0 & \text{otherwise} \end{cases}$$

In practice, take most frequently occurring n words (10,000 to 50,000) in training set, rather than manually pick 100 words.

Building a Spam Classifier

How to spend your time to build ML projects having low error ?

- Should we collect more data? *e.g.*
  - Use Project Honey Pot (*cf.* https://www.projecthoneypot.org/)
- Develop sophisticated features based on email routing information (*e.g.* from email header)
- Develop sophisticated features from message body *e.g.*
  - Treat 'discount' and 'discounts' as the same word?
  - Treat 'deal' and 'Dealer' as the same word?
  - Deal with punctuation (*e.g.* !)?
- Develop sophisticated algorithm to detect misspellings (*e.g.* m0rtage, med1cine, *etc.*)

run Error Analysis !

Question

Which of the following statements do you agree with? Circle all that apply.

(i) For some learning applications, it is possible to imagine coming up with many different features (*e.g.* email body features, email routing features, *etc.*). But, it can be hard to guess in advance which features will be the most helpful.

(ii) For spam classification, algorithms to detect and correct deliberate misspellings will make a significant improvement in accuracy.

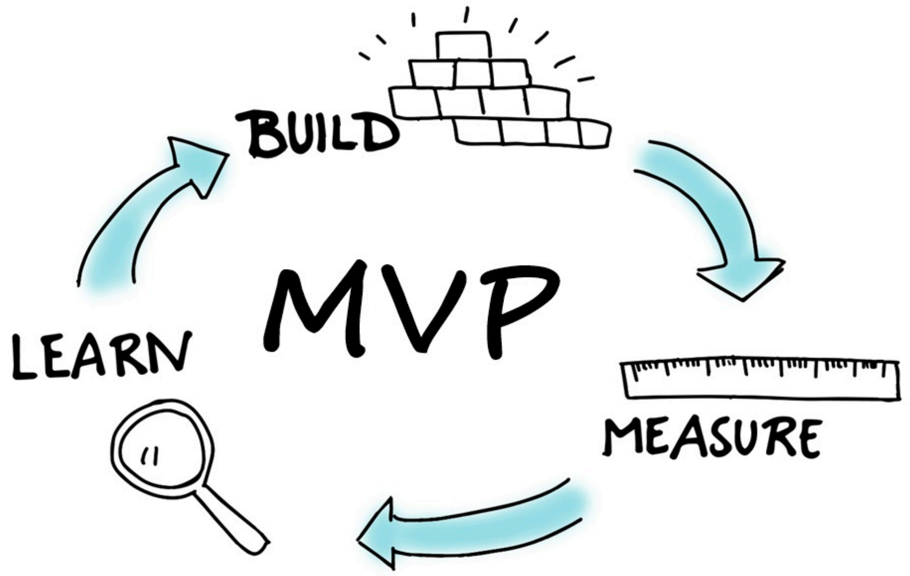
(iii) Because spam classification uses very high dimensional features (*e.g.* n = 50,000 if the features capture the presence or absence of 50,000 different words), a significant effort to collect a massive training set will always be a good idea.

(iv) There are often many possible ideas for how to develop a high accuracy learning systems; 'gut feeling' is not a recommended way to choose among the alternatives.

Error Analysis

Recommended Approach

1. Start with a simple approach that you can implement quickly. Implement it and test it on your cross-validation data.
2. Plot 'learning curves' to decide if more data, more features, *etc.* are likely to help (*i.e.* high bias or high variance).
3. **Error analysis:** Manually examine the examples (in cross validation set) that your algorithm made errors on. See if you spot any systematic trend in what type of examples it is making errors on.



Error Analysis (Example)

Suppose:

- You’ve built a spam classifier
- You’ve 500 examples in cross-validation set ($m_{\mathbf{cv}}$)
- Algorithm misclassifies 100 emails in cross-validation set
- Manually examine the 100 errors and categorize them based on:
  - what type of email it is
  - what features that can help it to classify them correctly

• Pharma:	12	• Deliberate misspelling: m0rtage, med1cine, *etc.*	5
• Replica / fake:	4	• Unusal email routing	16
• Steal password:	53	• Unusal (spamming) punctuation	32
• Other:	31		

Question

Why is the recommended approach to perform error analysis using the cross validation data used to compute $J_{\mathbf{cv}}(\theta)$ rather than the test data used to compute $J_{\mathbf{test}}(\theta)$?

(i) The cross validation data is usually large

(ii) This process will give a lower error on the test set.

(iii) If we develop new features by examining the test set, then we may end up choosing features that work well specifically for the test set, so $J_{\mathbf{test}}(\theta)$ is no longer a good estimate of how well we generalize to new examples.

(iv) Doing so is less likely to choosing an excessive number of features.

Evaluation of Learning Algorithm

- Should discount / discounts / discounted / discounting be treated as the same word ?
  - Can we use ‘stemming’ software (*e.g.* ‘Porter stemmer’)?
    - universe / university ✗

- Need numerical evaluation (*e.g.* cross validation error) of algorithm’s performance with and without something

Without stemming	**With stemming**
5% error	3% error

+ Distinguish uppercase & lowercase

3.2% error

Error metrics for skewed classes

Example of Skewed Classes

- Train logistic regression model $h_\theta(x)$
  - $y = 1$ if cancer; otherwise $y = 0$
- Find that you got 1% error on test set
  - 99% correct diagnose (Quite impressive !)

- Now, we find out 'only' 0.5% of patients have cancer
  - Hence, 1% error on test set is no longer impressive

(positive examples : negative examples)
is too close to one of the two !
***aka.* skewed classes**

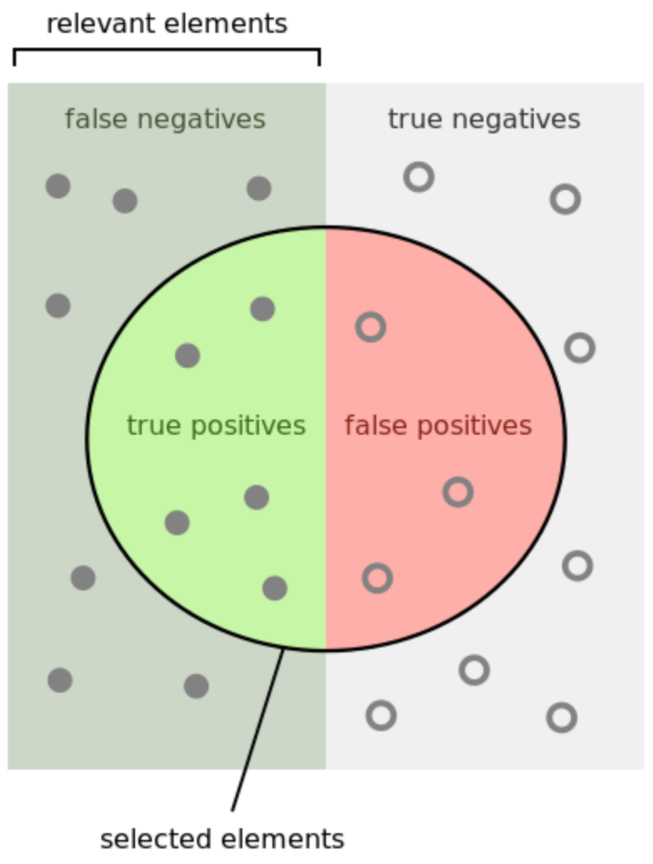
Even non-learning
can give less error rate !

```
def predict(x):
    y = 0 # always ignore x
    return y
```

99.5%
Accuracy

Precision & Recall

		Actual Class	
		1	**0**
Predicted Class	**1**	True Positive	False Positive
	0	False Negative	True Negative



Precision:
For all patients that we predicted $y = 1$,
what fraction actually has cancer?

$$\textbf{Precision} = \frac{\textbf{TP}}{\textbf{TP} + \textbf{FP}}$$

Recall:
For all patients that actually have cancer,
what fraction did we correctly detect as having cancer?

$$\textbf{Recall} = \frac{\textbf{TP}}{\textbf{TP} + \textbf{FN}}$$

Question

Your algorithm's performance on the test set is given below.
What is the algorithm's precision?

[✓ (i)] 0.8
(ii) 0.5
(iii) 0.08
(iv) 0.1

Recall = 0.5

		Actual class	
		1	0
Predicted class	1	80	20
	0	80	820

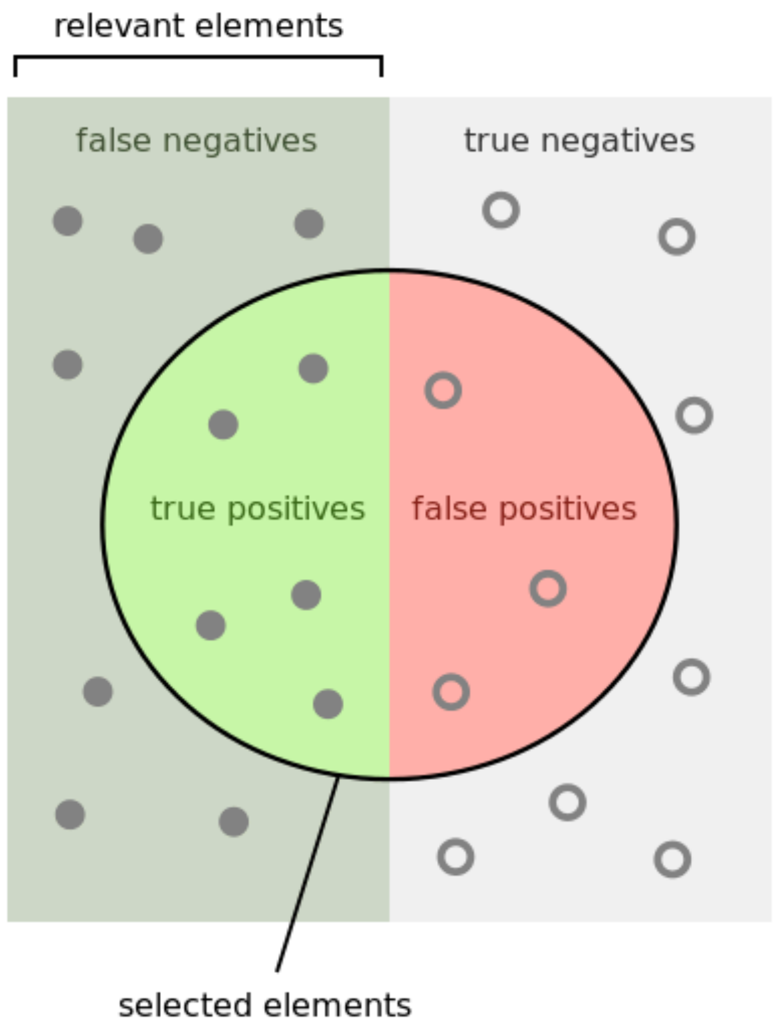
Trading Off
Precision & Recall

Motivating Example

Trading off precision and recall

Logistic regression: $0 \leq h_\theta(x) \leq 1$

- Predict 1 if $h_\theta(x) \geq 0.5$
- Predict 0 if $h_\theta(x) < 0.5$



Motivating Example

Trading off precision and recall

Logistic regression: $0 \leq h_\theta(x) \leq 1$

- Predict 1 if $h_\theta(x) \geq 0.5$ 0.7 0.3
- Predict 0 if $h_\theta(x) < 0.5$ 0.7 0.3

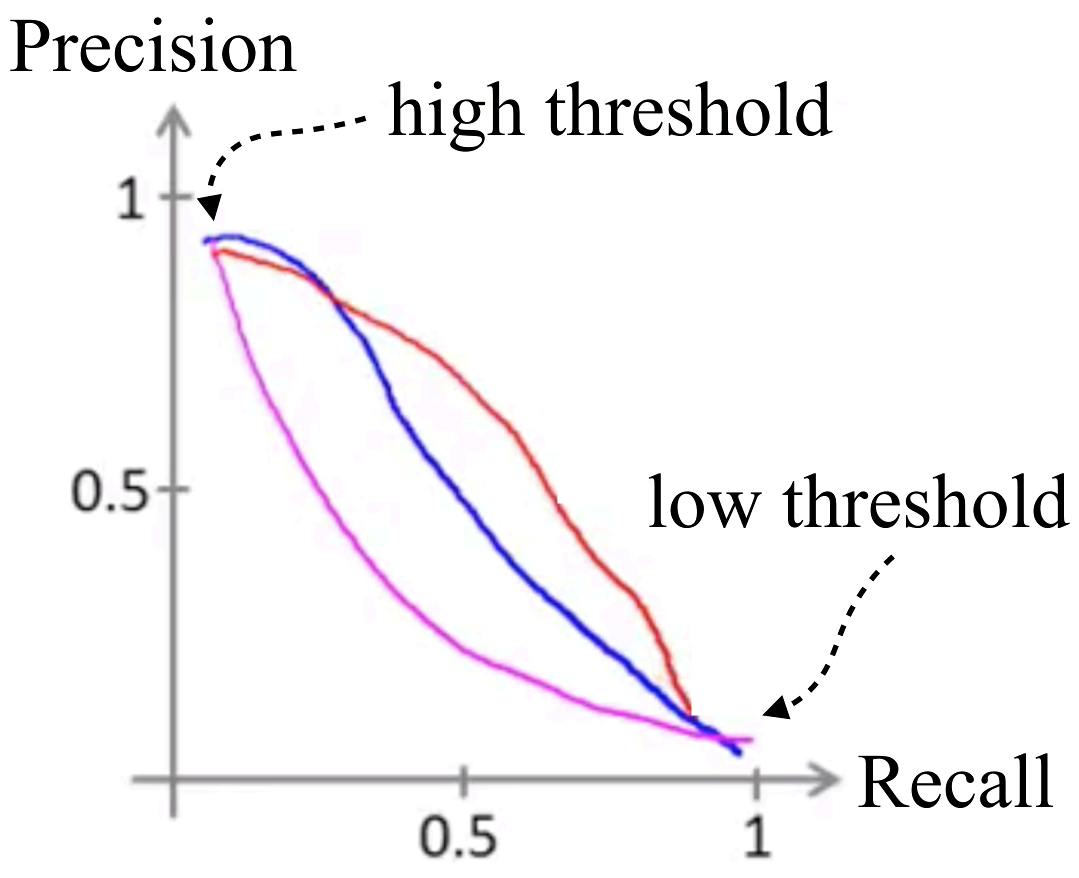
Suppose we want to predict $y = 1$ (*i.e.* cancer) only if we are very confident

- Higher precision, lower recall

Suppose we want to avoid missing too many cases of cancer (avoid false negatives)

- Higher recall, lower precision

More generally: Predict 1 if $h_\theta(x) \geq$ **threshold**



Is there a way to choose this threshold automatically?

F_1 Score (F Score)

How to compare precision / recall numbers?

	Precision(P)	Recall (R)	**Average**
Algorithm 1	0.5	0.4	0.45
Algorithm 2	0.7	0.1	0.4
Algorithm 3	0.02	1.0	0.51

(Algorithm 3 may be just predict $y = 1$ all the time)

Average: $\dfrac{P + R}{2}$

F_1 Score (F Score)

How to compare precision / recall numbers?

	Precision(P)	Recall (R)	~~Average~~	F_1 Score
Algorithm 1	0.5	0.4	~~0.45~~	0.444
Algorithm 2	0.7	0.1	~~0.4~~	0.175
Algorithm 3	0.02	1.0	~~0.51~~	0.0392

Average: $\cancel{\dfrac{P+R}{2}}$

(Algorithm 3 may be just predict $y = 1$ all the time)

F_1 Score: $2\dfrac{PR}{P+R}$

(why F Score is a good evaluation metric?)

- $P = 0$ **or** $R = 0 \Longrightarrow \mathbf{F}_1 = 0$
- $P = 1$ **and** $R = 1 \Longrightarrow \mathbf{F}_1 = 1$

Question

You have trained a logistic regression classifier and plan to make predictions according to:

- Predict $y = 1$ if $h_\theta(x) \geq$ **threshold**
- Predict $y = 0$ if $h_\theta(x) <$ **threshold**

For different values of the threshold parameter, you get different values of precision (P) and (R). Which of the following would be a reasonable way to pick the value to use for the threshold?

(i) Measure precision (P) and Recall (R) on the **test set** and choose the value of threshold which maximizes $(P + R)/2$

(ii) Measure precision (P) and Recall (R) on the **test set** and choose the value of threshold which maximizes $(2PR)/(P + R)$

(iii) Measure precision (P) and Recall (R) on the **cross validation set** and choose the value of threshold which maximizes $(P + R)/2$

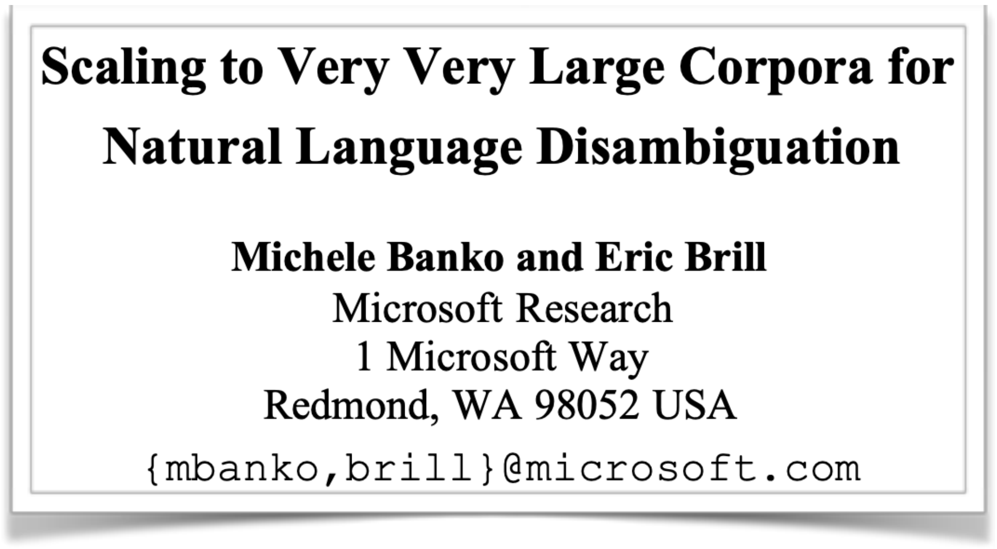
[✓ (iv)] Measure precision (P) and Recall (R) on the **cross validation set** and choose the value of threshold which maximizes $(2PR)/(P + R)$

Data for Machine Learning

Motivating Example

Classifying between confusable words *e.g.*
{to, two, too} and {then, than}

- For breakfast, I ate … eggs



Scaling to Very Very Large Corpora for Natural Language Disambiguation

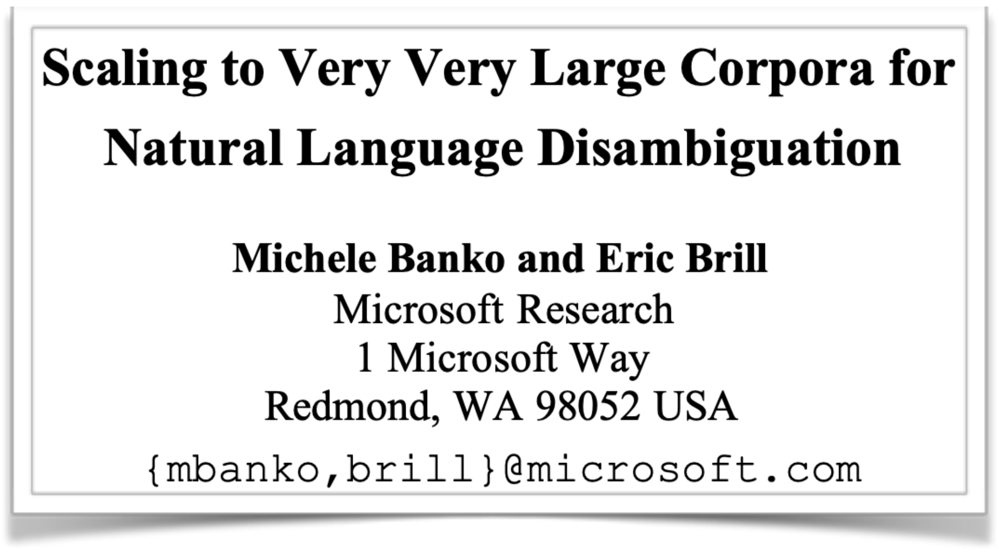
Michele Banko and Eric Brill
Microsoft Research
1 Microsoft Way
Redmond, WA 98052 USA
{mbanko,brill}@microsoft.com

Banko, M., & Brill, E. (2001, July). Scaling to very very large corpora for natural language disambiguation. In *Proceedings of the 39th annual meeting on association for computational linguistics* (pp. 26-33). Association for Computational Linguistics.

Motivating Example

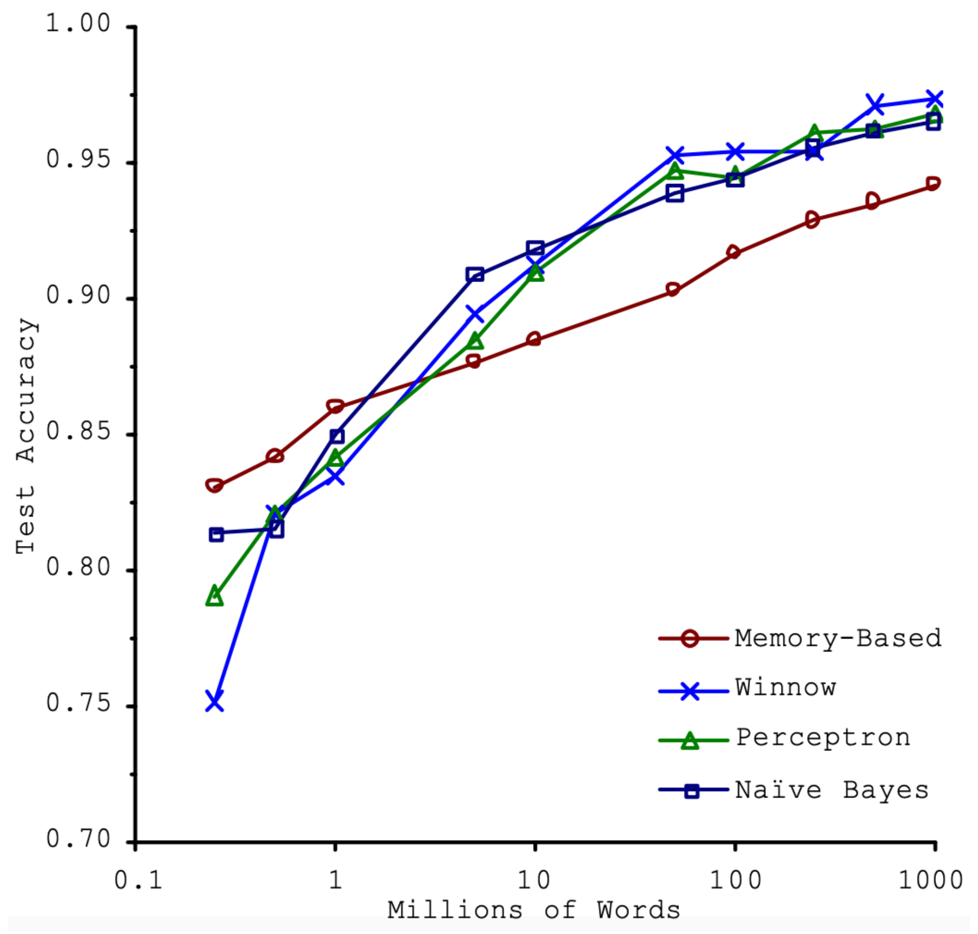
Classifying between confusable words *e.g.* {to, two, too} and {then, than}

- For breakfast, I ate … eggs



Scaling to Very Very Large Corpora for Natural Language Disambiguation

Michele Banko and Eric Brill
Microsoft Research
1 Microsoft Way
Redmond, WA 98052 USA
{mbanko,brill}@microsoft.com



"It's not who has the best algorithm that wins. It's who has the most data." — Andrew Ng

Under which conditions, this quote holds true ?

Banko, M., & Brill, E. (2001, July). Scaling to very very large corpora for natural language disambiguation. In *Proceedings of the 39th annual meeting on association for computational linguistics* (pp. 26-33). Association for Computational Linguistics.

Large Data Rationale

- Assume feature $x \in \mathbb{R}^{n+1}$ has sufficient information to predict y accurately

- Good example: For breakfast, I ate … eggs {two / to / too}
- Bad example: Predict housing price from only size (feet^2) and no other features

- Useful test case:
  - Given the input x, can a human expert confidently predict y?

Large Data Rationale

- Assume feature $x \in \mathbb{R}^{n+1}$ has sufficient information to predict y accurately

This is sometimes called 'low bias' algorithm

- Use a learning algorithm with many parameters (*e.g.*
  - logistic regression / linear regression with many features; or
  - neural network with many hidden units)

This leads to: $J_{\text{train}}(\theta)$ will be small.

This is sometimes called 'low variance' algorithm

- Use a very large training set *e.g.* much larger than #parameters
  - Hence, unlikely to overfit *i.e.*

$$J_{\text{train}}(\theta) \approx J_{\text{test}}(\theta)$$

Large Data Rationale

- Assume feature $x \in \mathbb{R}^{n+1}$ has sufficient information to predict y accurately

This is sometimes called 'low bias' algorithm

- Use a learning algorithm with many parameters (*e.g*.
  - logistic regression / linear regression with many features; or
  - neural network with many hidden units)

This leads to: $J_{\textbf{train}}(\theta)$ will be small.

This is sometimes called 'low variance' algorithm

- Use a very large training set *e.g*. much larger than #parameters
  - Hence, unlikely to overfit *i.e*.

$$J_{\textbf{train}}(\theta) \approx J_{\textbf{test}}(\theta)$$

If Low Bias and Low Variance, then training with a lot of data would HELP !

Question

Having a large training set can help significantly improve a learning algorithm's performance. However, the large training set is **unlikely** to help when:

(i) The features x do not contain enough information to predict y accurately (*e.g.* predicting a house's price from only its size), and we are using a simple learning algorithm *e.g.* logistic regression

(ii) We are using a learning algorithm with a large number of features (*i.e.* one with 'low bias')

(iii) The features x do not contain enough information to predict y accurately (*e.g.* predicting a house's price from only its size), even if we are using a neural network with a large number of hidden units

(iv) We are not using regularization (*e.g.* the regularization parameter $\lambda = 0$)