Data Compression Models

for

Prediction and Classification

Gordon CORMACK[1]
University of Waterloo, Waterloo, Canada

Abstract. The crux of data compression is to process a string of bits in order, predicting each subsequent bit as accurately as possible. The accuracy of this prediction is reflected directly in compression effectiveness. Dynamic Markov Compression (DMC) uses a simple finite state model which grows and adapts in response to each bit, and achieves state-of-the art compression on a variety of data streams. While its performance on text is competitive with the best known techniques, its major strength is that is lacks prior assumptions about language and data encoding and therefore works well for binary data like executable programs and aircraft telemetry.

The DMC model alone may be used to predict any activity represented as a stream of bits. For example, DMC plays "Rock, Paper Scissors" quite effectively against humans. Recently, DMC has been shown to be applicable to the problem of email and web spam detection -- one of the best known techniques for this purpose. The reasons for its effectiveness in this domain are not completely understood, because DMC performs poorly for some other standard text classification tasks. I conjecture that the reason is DMC's ability to process non-linguistic information like the headers of email, and to predict the nature of polymorphic spam rather than relying fixed features to identify spam.

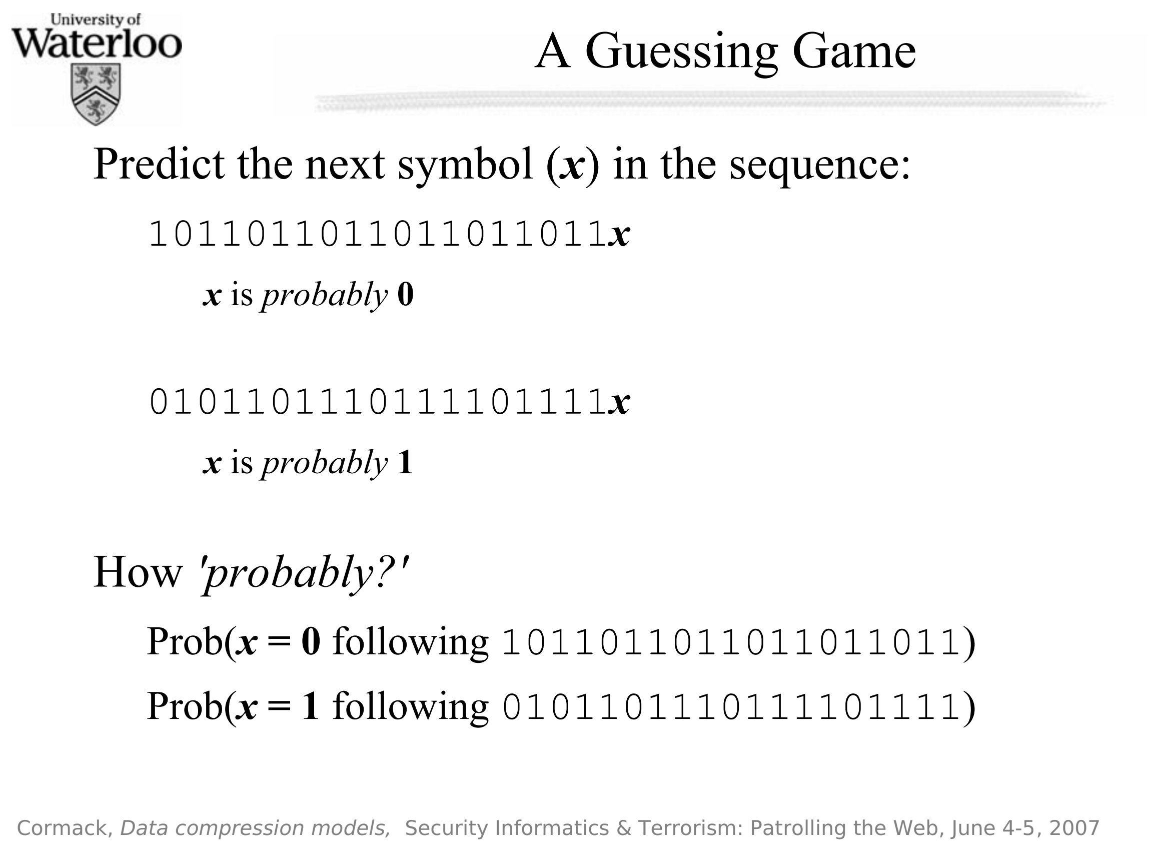
In this presentation I describe DMC and its application to classification and prediction, particularly in an environment where particular patterns of data and behavior cannot be anticipated, and may be chosen by an adversary so as to defeat classification and prediction.

Keywords. Prediction, Classification, Markov Model, Compression, Screening, Spam

I'd like to play a guessing game with you. I'll start with a very simple version. I'll give you a sequence of numbers, zeros and ones, and I'll stop at some point, and you tell me what the next one is. So, if we look at the first sequence [Figure 1], one zero, one one zero, one one zero one one zero one one, and I ask you what the next one is, probably you'll tell me it's zero. I'm going to make things a little harder though. I'm going to say, "what odds will you give me that it's zero?" And so those are the two basic problems, "guess what the answer is" and "give me odds." And of course, this string may never have happened before but I still want you to guess. Here's another string: the second one [in Figure 1] is a little bit more complicated, zero one zero, one one zero, one one one zero, one one one one zero and so on. What's the next? Again, we can say it's probably one and I'll say that's not good enough. I want to know "how

[1] Corresponding Author: Gordon Cormack, David R. Cheriton School of Computer Science, 2502 Davis Centre, University of Waterloo, Waterloo, Ontario N2L 3G1, Canada; Email: cormack@cormack.uwaterloo.ca

probably” it is one. And so, in a nutshell, I want to know, “given a string what is the probability distribution for what the next symbol might be?”



University of Waterloo

A Guessing Game

Predict the next symbol (***x***) in the sequence:

`1011011011011011011`***x***

x is *probably* **0**

`0101101110111101111`***x***

x is *probably* **1**

How *'probably?'*

Prob(***x*** = **0** following `1011011011011011011`)

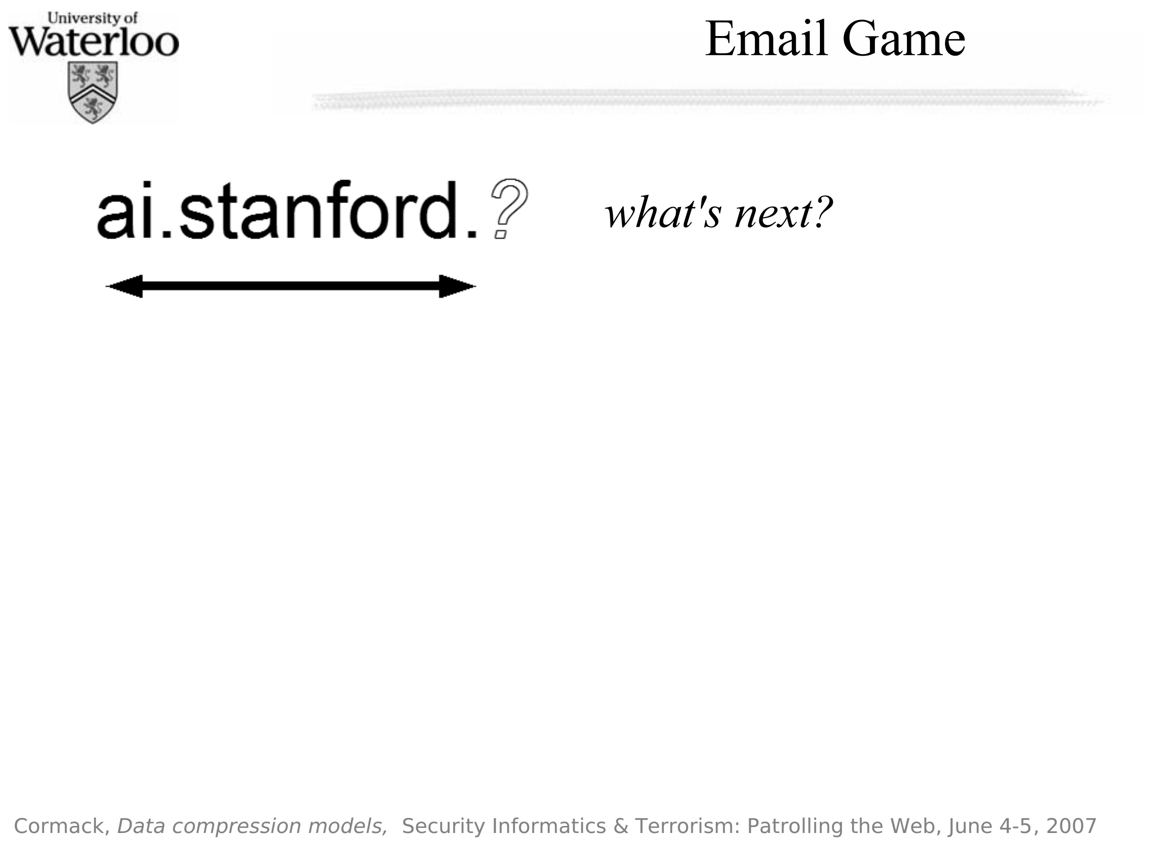
Prob(***x*** = **1** following `0101101110111101111`)

Cormack, *Data compression models,* Security Informatics & Terrorism: Patrolling the Web, June 4-5, 2007

Figure 1. A simple prediction game

Here [Figure 2] is another version of the game. I have a number of email messages from an inbox, let’s say I have 100,000 of them and I pick one of the ones that happens to have “ai.stanford.” in it. What’s the next symbol after “ai.stanford.?” Any guesses? An E. Well, one of the messages has an E, but in fact your answer was underspecified, because in the data set there are actually two forms of E, lowercase “e” and uppercase “E.” Any guesses which is the more common? Well, if you guessed lowercase “e,” you’d be wrong. In this particular corpus there was one lowercase “e” and there were 509 uppercase E’s [Figure 3]. And not only that, I’m going to make an observation that was not part of the game: All 509 of the uppercase E’s were in spam messages. And the one lowercase E was the one legitimate message. So what further predictions might we make from this? First of all, if we want to know, “what’s spam or not,” capital E is a very good predictor of spam, at least in this data set. And we might be tempted to say 509 to 1 odds. That would just be an approximation, but it’s a common way of doing approximation, you count proportions of things you’ve seen before and assume that that’s going to predict a proportion that you’ll see in advance and therefore the probability.

Now, Paul [Kantor] don’t bother with your question about, “you know, the world isn’t a stochastic place?” It isn’t, but in this case it’s modeled fairly well by this. The point is if you see a lowercase E, you’ve seen a lowercase E once and it was not a spam message, so does that mean that we can predict from seeing another lowercase E that it’s not a spam message? Our odds are one to zero, that’s infinite. So, we are 100% sure. Well, that would be a really bad assumption. And probably wrong, from our intuition as to what this data means. Here is an even worse assumption: If we see “ai.stanford” in an email message the chances that it’s spam are 509 to 1. That’s almost certainly a specious assumption inferred from this particular data. But it’s not completely obvious just from looking at this example without any intuition.



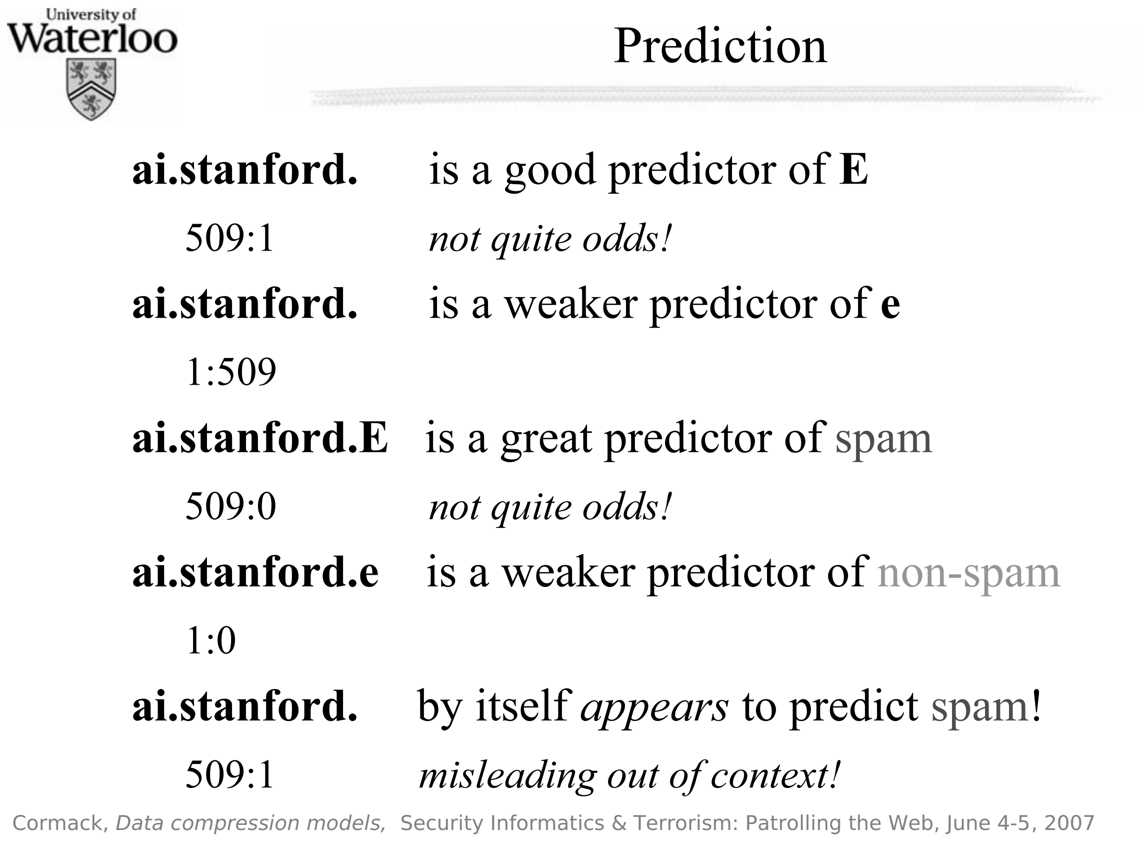
Email Game

ai.stanford.?

what's next?

Cormack, *Data compression models,* Security Informatics & Terrorism: Patrolling the Web, June 4-5, 2007

Figure 2. Guess the next symbol



Prediction

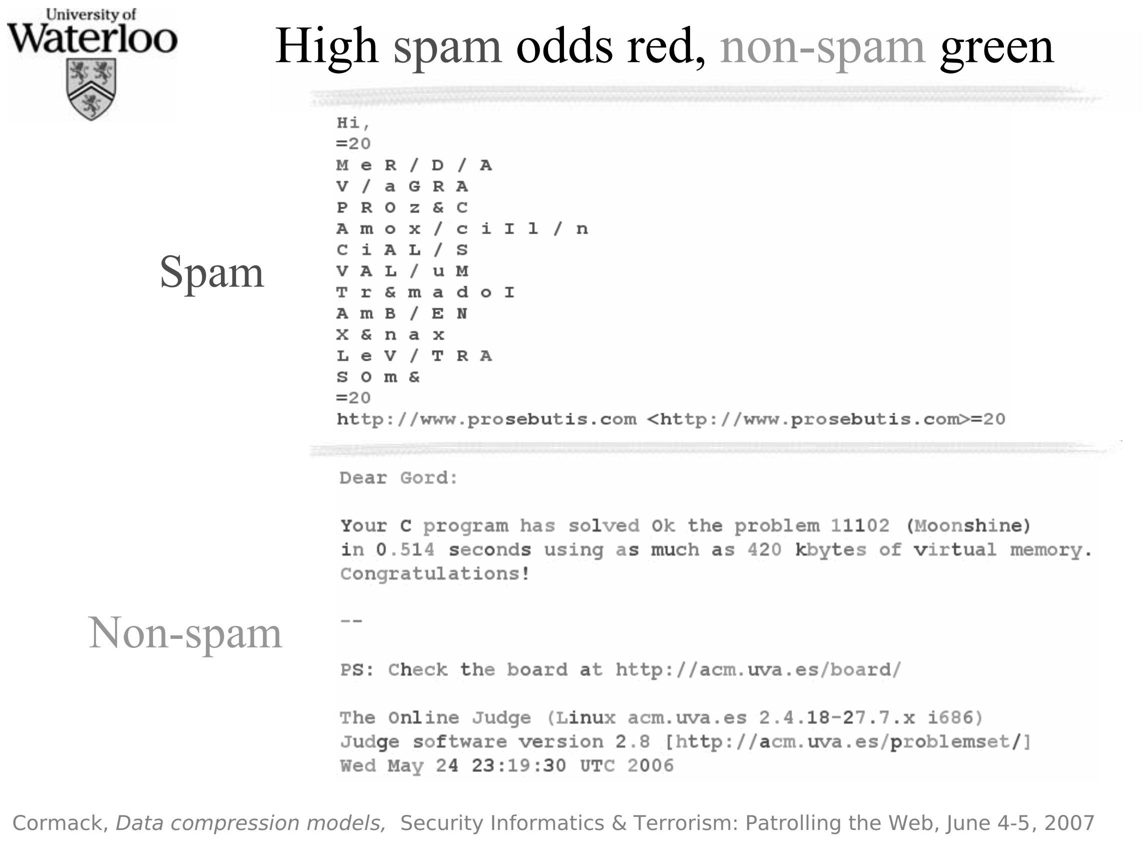
ai.stanford.	is a good predictor of **E**
509:1	*not quite odds!*
ai.stanford.	is a weaker predictor of **e**
1:509	
ai.stanford.E	is a great predictor of spam
509:0	*not quite odds!*
ai.stanford.e	is a weaker predictor of non-spam
1:0	
ai.stanford.	by itself *appears* to predict spam!
509:1	*misleading out of context!*

Cormack, *Data compression models,* Security Informatics & Terrorism: Patrolling the Web, June 4-5, 2007

Figure 3. Observed frequencies of "e" and "E"

Here is another version of the game. Well, this is basically the same game. I have two email messages, or at least fragments of email messages here and I want to find the spam or I want to find the non-spam. Just to make this a little bit more fun -- as they do in game shows, sometimes you have to make a weird noise or say a strange word when you have the answer -- if you see spam you have to color it red and if you see non-spam you have to color it gray. So here [Figure 4] is actually what my program that plays this game does. It doesn't completely color the messages black or gray; it colors the bits of the message that it thinks are spammy black and bits that are non-spammy, gray. And it does a great job on the spam message. One of the reasons it can do a great job on the spam messages is that somebody architected this message to try to beat spam filters. They split it up so that there are none of these features or "bag of words" things that all of the machine learning people who have talked here already love so much. There just aren't any in this message. And that will beat somebody who depends on putting things in words, but it certainly doesn't beat this particular technique. It [the technique] is not so unequivocal on the non-spam message. But again, if you look at it "from a distance" it's more gray than it is black and this supports the view that one should combine all sources of evidence in making a decision.

You have to step back and look at the whole picture; the whole picture is pretty clear. The first one was identified as spam because it has this chopped up feature. In the non-spam my name was non-spammy and there were various other words and bits of words that indicated non-spamminess.



High spam odds red, non-spam green

```
Hi,
=20
M e R / D / A
V / a G R A
P R O z & C
A m o x / c i I l / n
C i A L / S
V A L / u M
T r & m a d o I
A m B / E N
X & n a x
L e V / T R A
S O m &
=20
http://www.prosebutis.com <http://www.prosebutis.com>=20
```

```
Dear Gord:

Your C program has solved Ok the problem 11102 (Moonshine)
in 0.514 seconds using as much as 420 kbytes of virtual memory.
Congratulations!

--

PS: Check the board at http://acm.uva.es/board/

The Online Judge (Linux acm.uva.es 2.4.18-27.7.x i686)
Judge software version 2.8 [http://acm.uva.es/problemset/]
Wed May 24 23:19:30 UTC 2006
```

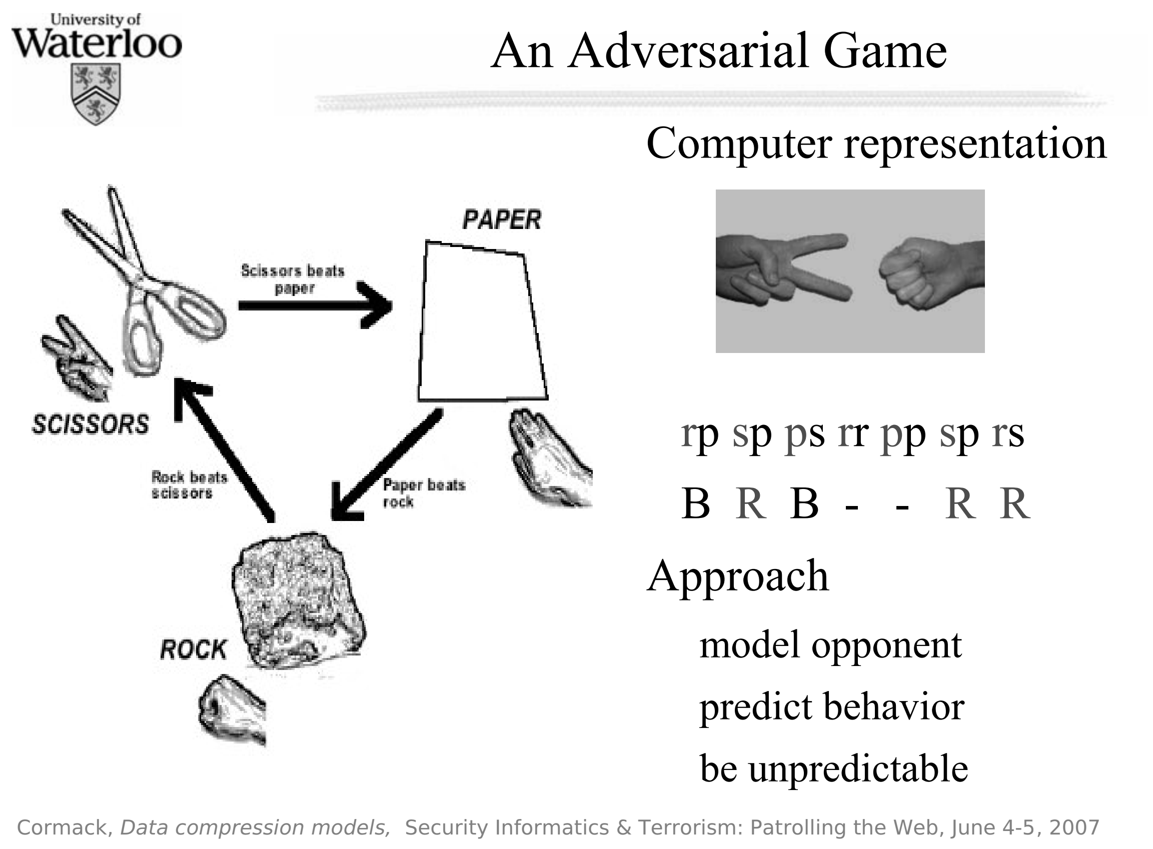
Cormack, *Data compression models,* Security Informatics & Terrorism: Patrolling the Web, June 4-5, 2007

Figure 4. Spam coloring game

I'm going to talk very briefly about a method -- the method I use to do this coloring. But I'm also going to talk about that in context of … as an academic I love to play games, …. but really this is in support of a genuine application. We should always be asking, "if I learn to play this game really well does tt really help to solve the spam filtering problem?" So I need to look at a bigger context and I need to ask if what I am measuring about this actually predicts whether this is solving the email reading problem.

I'm also going to talk about other applications, or other versions of this game. But first, before I talk about the methods, I want to give you one more game. This is an adversarial game, it's called Rocks-Scissors, Paper-Scissors or Roshambo, and it's played with a rock and paper and scissors. Well, it's actually not played with a rock and paper and scissors, it's played with hand gestures that sort of look like these things and two people simultaneously chose one of these things. The game has a cyclical dominance graph [Figure 5].Iif you get dominated you lose and if you dominate you win. So, the key to the game is actually to guess what the other person is going to play at the same time as you are and then to beat them. And of course, we are not going to use hands here; we're going to write down an R for rock, and a P for paper and so on. And then we'll pair these things up, but together they're just a string of characters [Figure 5, bottom right] a spam message [Figure 4] So what we want to do once again is to read a bunch of stuff that's been played and then we want to say, "well, what's the next move my opponent is going to make,?" and then I want to beat him. How do I beat do that? I consider the move that he is most likely going to make and I play the move that will beat that. But in fact it's not quite as simple as that, because I also am concerned very much about my opponent being able to predict *my* behavior.

So I have a big trade-off between telling my opponent what I know and beating my opponent on this particular round. In short, there is, the cost (well, benefit) of winning this round versus the cost of what I communicate to my opponent, not only what I communicate to my opponent but what I communicate perhaps to a third party observer. If this is a "Rock, Paper, Scissors" tournament I'm playing, the person I'm up against next is watching how I play, trying to decide how good I am. And others may be watching me as well. So, I should really introduce enough noise so that I only just play well enough to beat my opponent; or I only play well enough to do what I need to qualify for the next round. In any event, there are these two trade-offs but that doesn't really change the fact that underlying the game, the main thing we need is to predict the opponent's next move with an odds ratio or with odds.


University of Waterloo
PAPER
Scissors beats paper
SCISSORS
Rock beats scissors
Paper beats rock
ROCK

An Adversarial Game

Computer representation

rp sp ps rr pp sp rs

B R B - - R R

Approach

model opponent

predict behavior

be unpredictable

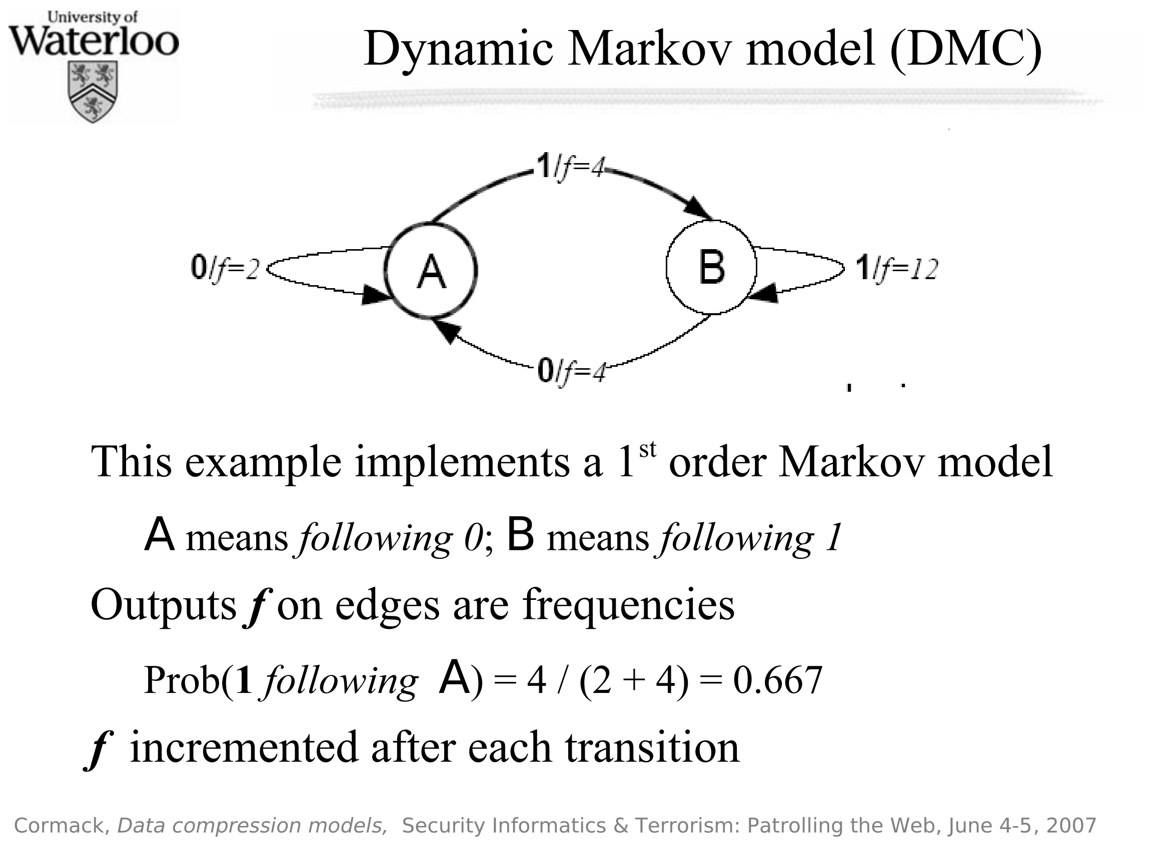
Cormack, *Data compression models,* Security Informatics & Terrorism: Patrolling the Web, June 4-5, 2007

Figure 5. Rock-paper-scissors game

All right, now let's go back to the main game of which these are all instances [Figure 1]. We have a sequence of symbols -- without loss of generality, they're all bits -- and we want to predict the next one. The particular method that I am going to talk about today is called Dynamic Markov Modeling, and it was something I first did in the 1980's (actually for data compression). It's an extremely simple algorithm. Bit I am going to oversimplify it even further; (If you want the code, not the code for the game, but the code for the model it's on the Web right now; search for "dmc.c"; if that's not enough put in my name). The entire compressor is 250 lines of code and half of that, approximately, is the model. So it really is as trivial as it seems. The way you work this model is: you put your finger on A [Figure 6] and then if you see a 1 you move your finger to B, and if you see a zero you move your finger back to A and so on. And you do this in sequence. Then at any given time,when you follow one of the arrows you increment the count on that arrow. So finally, the count on the two arrows that your finger might follow -- "one" and the "zero" – give the odds estimate; the ratio of the counts is the odds estimate.

It is that simple. Now there's nothing special about this, this is a very trivial Markov model; they've been talked about already. Where it gets a little bit more interesting is that after a while you say, "wait, I've had my finger on B a lot". "Not only that, I've gone from A to B a lot." And so whenever, when you go from A to B, when you hit some threshold you say, "oh, I think I'm going to make a new version of B." So you clone B and you divert some of the traffic from A, in particular, all the traffic from A to B [Figure 7]. You divert to your new cloned version of B. And all the other traffic you leave it where it was. So you start a “bypass” for this particular path. And what happens is, this grows and grows and grows and eventually yields a big model that is a *variable order* Markov model. You don’t have to worry about back-off probabilities and everything.

There are a few arbitrary constants such as what you initialized these weights to. But when you actually do this cloning operation, you simply split the traffic in proportion to how often you visited it from this place or state, versus from other places. So anyway, that’s DMC; that is as much as I’m going to say about the actual model other than it plays exactly our game [Figure 8].



1/*f*=4
0/*f*=2
A
B
1/*f*=12
0/*f*=4

University of Waterloo

Dynamic Markov model (DMC)

This example implements a 1st order Markov model

A means *following 0*; B means *following 1*

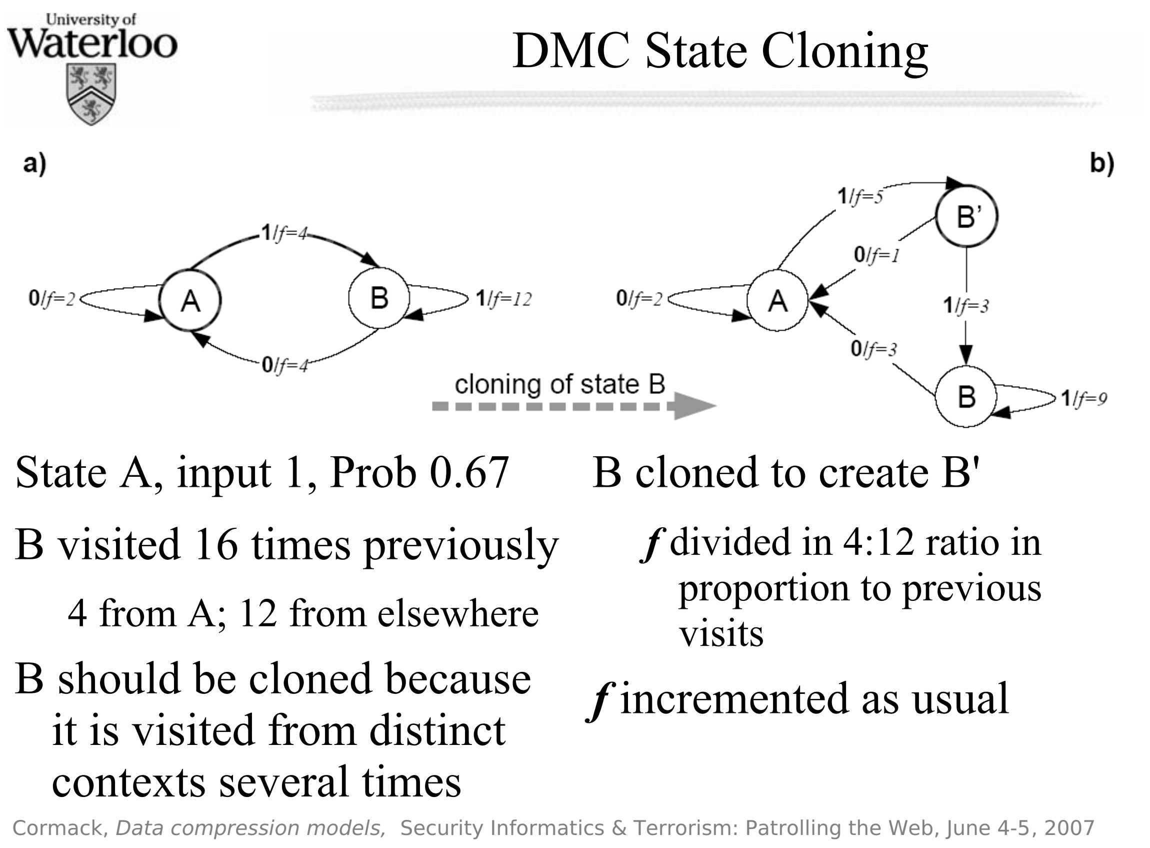
Outputs ***f*** on edges are frequencies

Prob(**1** *following* A) = 4 / (2 + 4) = 0.667

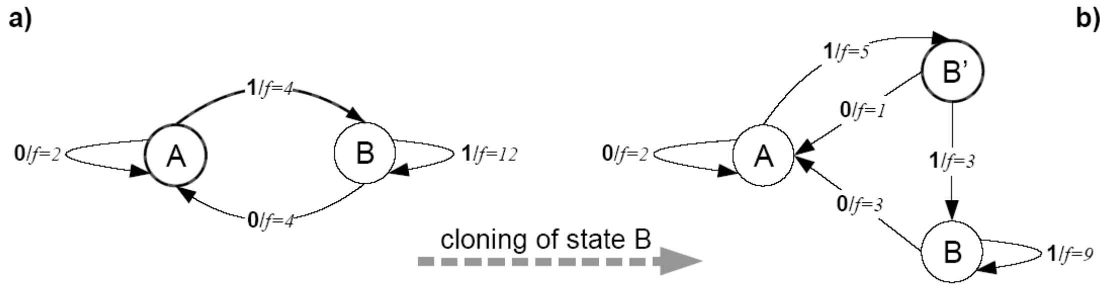
f incremented after each transition

Cormack, *Data compression models,* Security Informatics & Terrorism: Patrolling the Web, June 4-5, 2007

Figure 6. A Markov model



DMC State Cloning

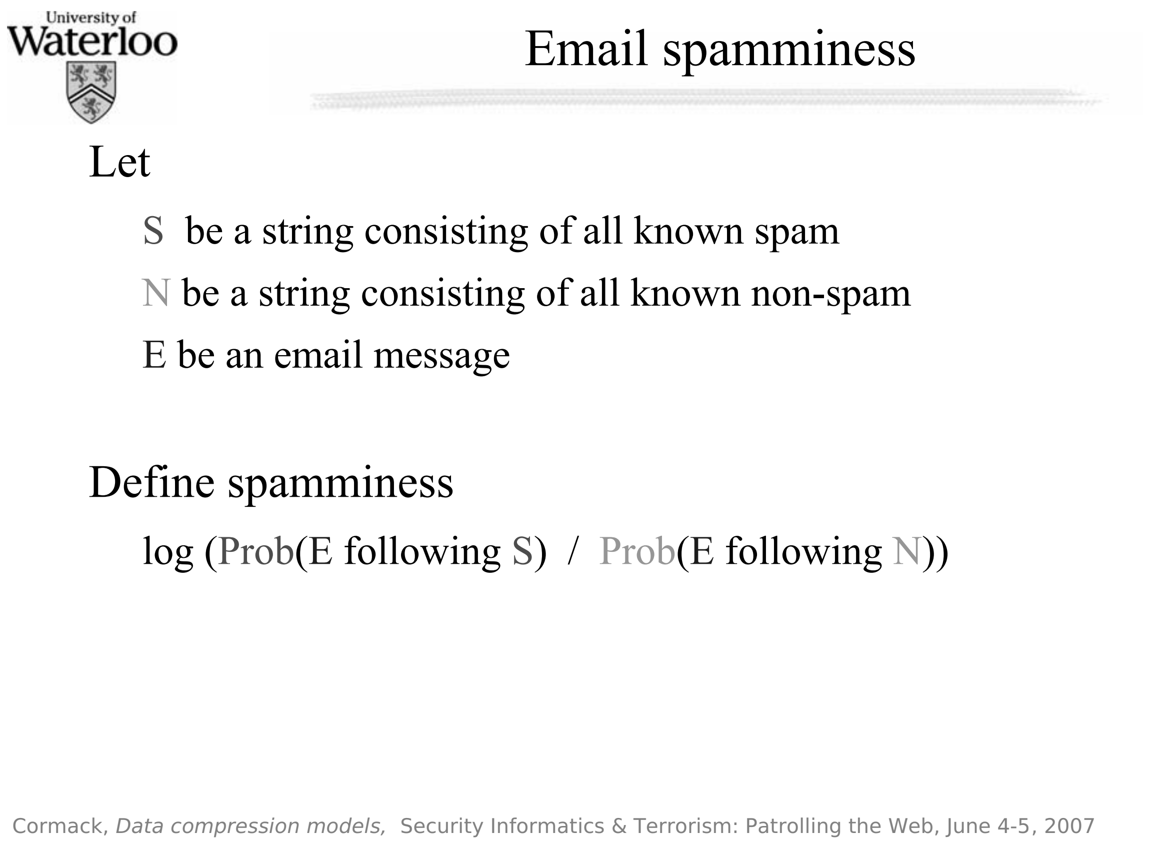


- State A, input 1, Prob 0.67
- B visited 16 times previously
  - 4 from A; 12 from elsewhere
- B should be cloned because it is visited from distinct contexts several times

- B cloned to create B'
  - ***f*** divided in 4:12 ratio in proportion to previous visits
- ***f*** incremented as usual

Cormack, *Data compression models,* Security Informatics & Terrorism: Patrolling the Web, June 4-5, 2007

Figure 7. Dynamic Markov model



Email spamminess

Let

S be a string consisting of all known spam

N be a string consisting of all known non-spam

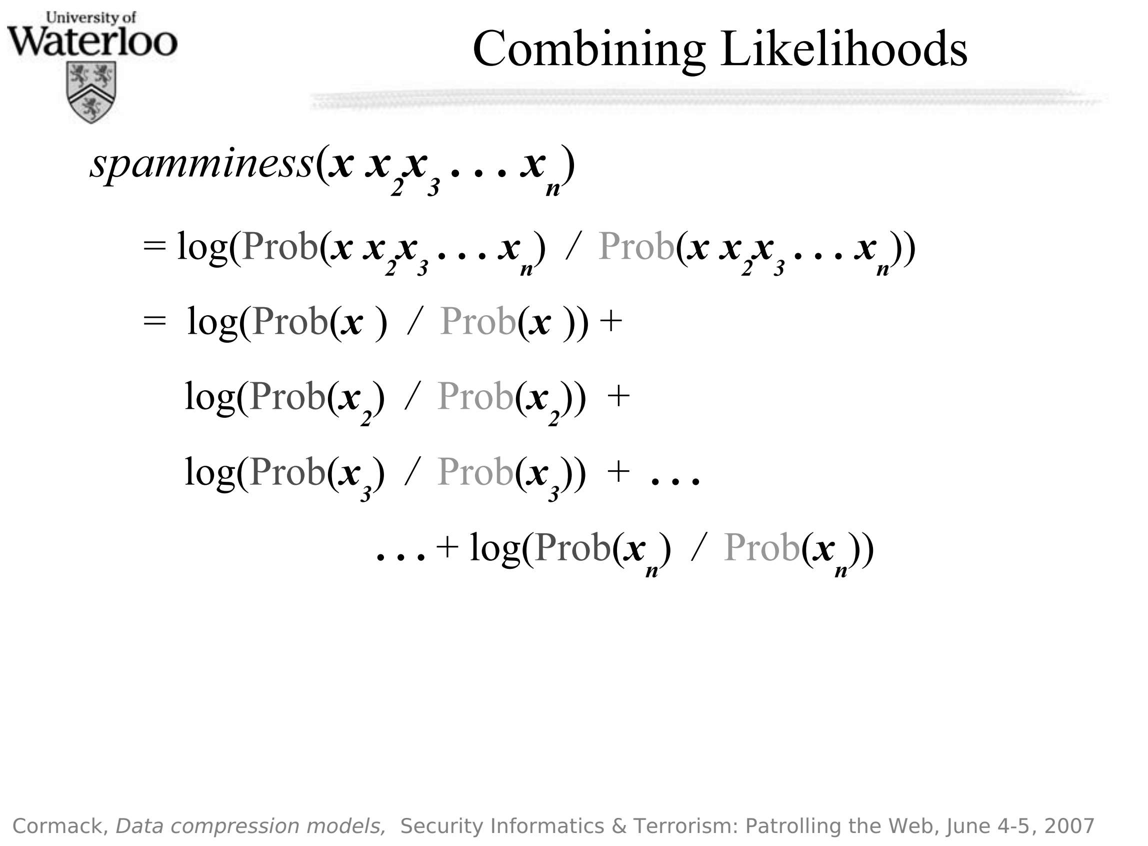
E be an email message

Define spamminess

log (Prob(E following S) / Prob(E following N))

Cormack, *Data compression models,* Security Informatics & Terrorism: Patrolling the Web, June 4-5, 2007

Figure 8. Spam to Ham likelihood ratio



Combining Likelihoods

$$spamminess(\boldsymbol{x}\,\boldsymbol{x}_2\boldsymbol{x}_3\ldots\boldsymbol{x}_n)$$

$$= \log(\mathrm{Prob}(\boldsymbol{x}\,\boldsymbol{x}_2\boldsymbol{x}_3\ldots\boldsymbol{x}_n)\ /\ \mathrm{Prob}(\boldsymbol{x}\,\boldsymbol{x}_2\boldsymbol{x}_3\ldots\boldsymbol{x}_n))$$

$$= \log(\mathrm{Prob}(\boldsymbol{x}\ )\ /\ \mathrm{Prob}(\boldsymbol{x}\ )) +$$

$$\log(\mathrm{Prob}(\boldsymbol{x}_2)\ /\ \mathrm{Prob}(\boldsymbol{x}_2))\ +$$

$$\log(\mathrm{Prob}(\boldsymbol{x}_3)\ /\ \mathrm{Prob}(\boldsymbol{x}_3))\ +\ \ldots$$

$$\ldots + \log(\mathrm{Prob}(\boldsymbol{x}_n)\ /\ \mathrm{Prob}(\boldsymbol{x}_n))$$

Cormack, *Data compression models,* Security Informatics & Terrorism: Patrolling the Web, June 4-5, 2007

Figure 9. Combining likelihood ratios

Now, if you want to predict something bigger than az single bit it's easy, you just predict all the bits in sequence. And then it doesn't really make much difference but you can either average the predictions or you can multiply the predictions together or you can sum them. Anyway, you can combine them in a number of ways, and actually for the purpose of this presentation it doesn't matter how you combine them. Basically, you are taking the average prediction over the entire string, that's how you are predicting an email. That's essentially what Figure 9 says. So what we do with email now is this, if we are trying to predict spam or non-spam, we take all the spam we've ever seen and stick them all together into a sequence and then predict how likely it is that the new message that we are trying to judge, would occur in a list of spam messages. And we also predict how likely it is to occur in a list of non-spam messages and that's our odds. The method is absolutely simple as that; we take the ratio of those, well, actually it's convenient to take the log of the odds, just so when we sum these logarithms of ratios we get something meaningful.

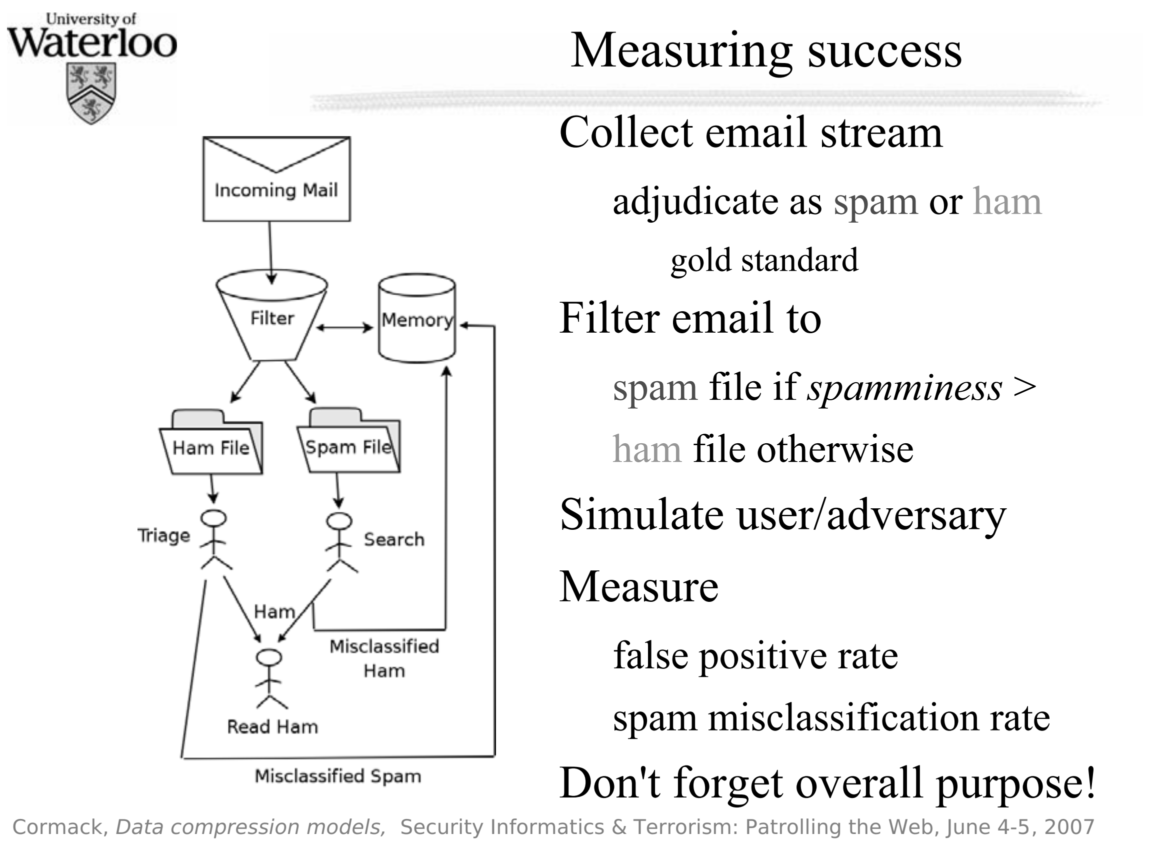
But the question is: how well does this work? Well, I can tell you "it works great". But let me do two things. First of all, I want to talk about measures of how well it actually plays the game. To do this, I want to show you a little picture of the context. In the context if this conference I what to stress that I think we should always be doing this -- you know, drawing a stick diagram like this [Figure 10] asking "where does this mathematical game actually fit in the overall picture?"

For the case of email the typical email reader looks something like this, there is an incoming stream of mail, the filter puts it into a mail file or into a quarantine file, and you routinely read your mail file. So it annoys you if there's spam in the mail file. If you feel that there's some (psychological) benefit to venting your emotional state, you might even yell at the filter and say, "how could you misclassify this?" and if the filter is smart it will say "well, I'm sorry I'll try not to do that again". In fact, this is what the DMC model can do; it can grow and clone and learn the kinds of mail that are spam and non-spam.

Now, I should mention that no user will ever tell you the correct classification for every mail. But they might, if the errors are rare enough, tell you about the errors that they notice or at least some substantial fraction of the errors that they notice. So what you have to do is to assume (by default) that you did it right unless you hear otherwise; this is an example of *bootstrapping* or *positive feedback*. If you're right often enough it works. The quarantine file is more difficult because, in general, the user doesn't look at it. And the better your filter is the less likely the user is to look at it. Of course, if you are perfect that's fine; but if you are not perfect there might be some misclassified mail. In general, looking in the spam file is a completely different information retrieval task from reading actual email. The quarantine is something that the user has to search. Maybe she's bored and just wants to look through it and see if there are any needles in the haystack, i.e., errors. Or maybe, she just booked an airline flight and didn't receive the message and says, "I bet there's a message from Expedia in there." You have to put these factors all together in order to figure out what the *cost*, the *downside* of a misclassification is.

It didn't really cost her that much to lose that travel agent booking because she was expecting it; even though it was valuable information, she was expecting it and she knew exactly how to get it. On the other hand, say, Paul sent me a message the day before my flight saying "the workshop is cancelled; don't come!" and it went into my spam file, that could have disastrous consequences. Happily, , that happens to be the kind of message that this method would be very unlikely to get wrong because he and I had corresponded already and the filter would have had ample opportunity to develop a positive attitude about this kind of message.

To sum up, when we measure misclassification rates they can be extremely misleading because 1 in 100 errors or 1 in 1000 errors could be disastrous. If errors are one in 1,000, but they are all critical messages that I wasn't going to get by some alternate channel got lost, that would be unacceptable. On the other hand, if the advertising for my frequent flyer plan gets lost, who cares?



Measuring success

Collect email stream

adjudicate as spam or ham

gold standard

Filter email to

spam file if *spamminess* >

ham file otherwise

Simulate user/adversary

Measure

false positive rate

spam misclassification rate

Don't forget overall purpose!

Cormack, *Data compression models,* Security Informatics & Terrorism: Patrolling the Web, June 4-5, 2007

Figure 10. Context for the spam coloring game

So, what I want to do is, I want to measure error rates. And what I'm *not* going to do is to talk about precision or recall or F-measure. I can't, in my mind, get any picture for what precision recall or F-measure would mean in terms of this overall picture; it wouldn't mean anything to me. On the other hand, I've already mentioned measurements that do mean something even though they may not be strictly proportions. That is, the number of spam, or proportion of spam misclassified, and the proportion of non-spam misclassified. So we can should one against the other because I have a "paranoia threshold" and I can vary it from, "no paranoia" to "lots of paranoia" and just plot one measure against the other. Now this is plotted on the log odds scale [Figure 11]; otherwise it is exactly the ROC curve that Paul talked about earlier and the black line is what our DMC does and all the other lines are what the best other known spam filters do. The number, area under the curve of the ROC, is very high – perhaps 0.999 – so, for whatever reason in this particular application, classifiers are two or three orders of magnitude more accurate than they are for others like Reuters or 25 newsgroups.

Not only that, DMC is pretty mediocre for some of those standard text classification tasks but it really works in this application. I want to go back if I can to Figure 11. If we look at this graph, a reasonable place to look at this graph would be on the first X tick, at 0.1%; that is, one in one thousand good emails misclassified. And if you look at the curve up here at x=0.1% you find that the filter gives slightly less than one percent spam misclassification, so it gets rid of more than 99% of spam while only misclassifying about one in a thousand good messages. Again, useful to think about 1 in a thousand just as a proportion because these are rare occurrences, so let's look at them for some spam filters.

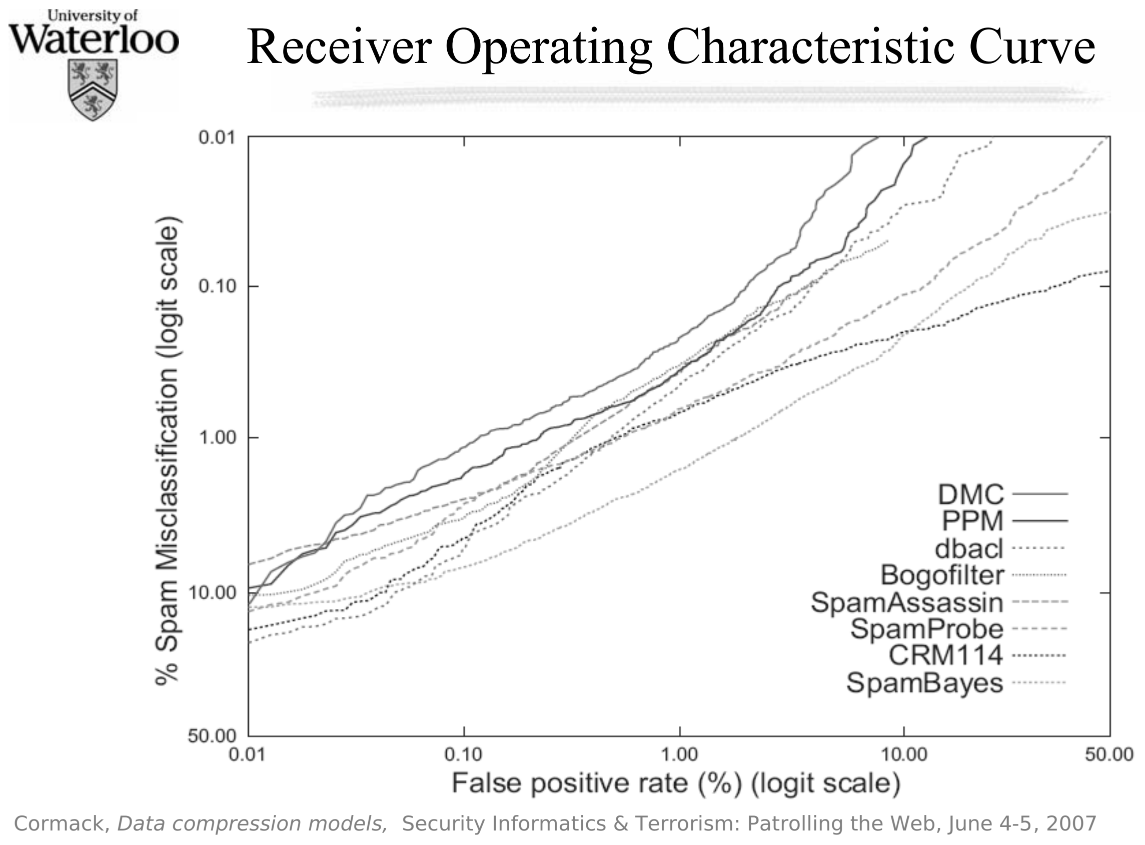
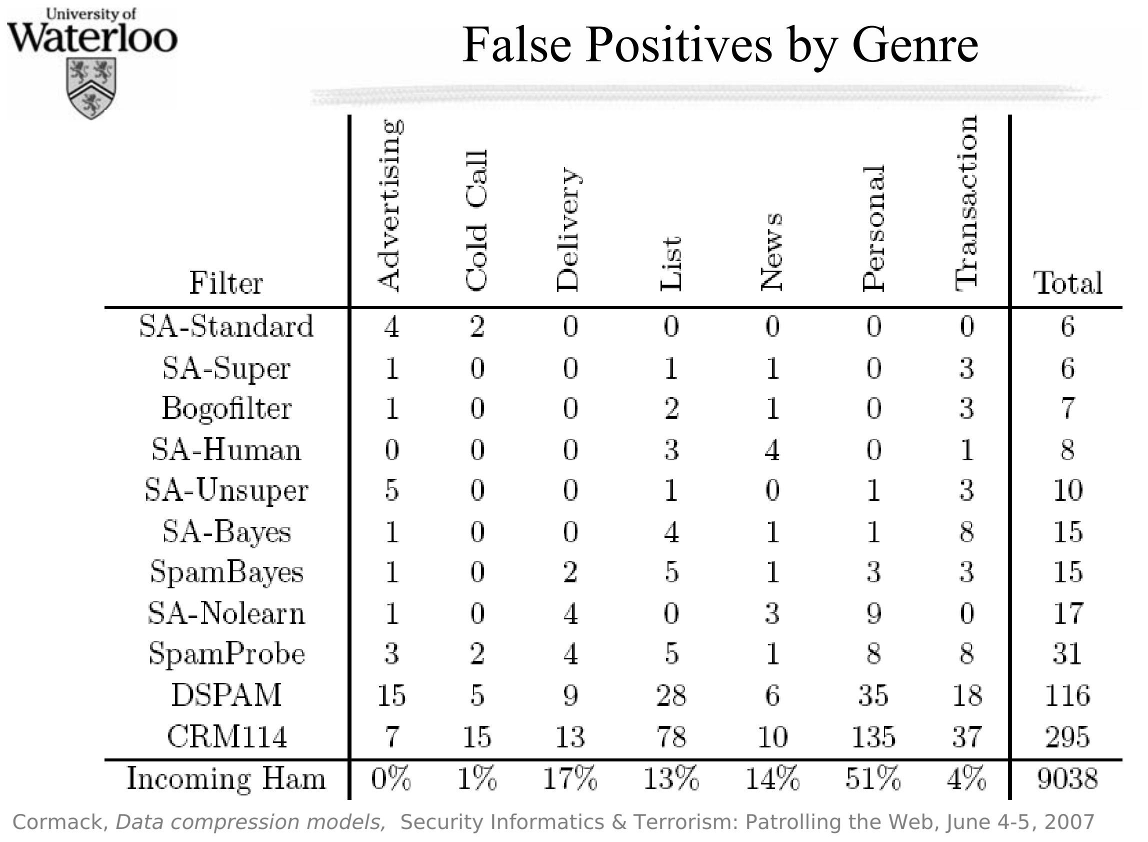


Figure 11. Quantitative results for spam filters [1]

Unfortunately, my spam filter isn't in this table [Figure 12] but this still illustrates the idea. Of 9000 good email messages the best filters here misclassified 6 or 7. And if you categorize these by genre you'll see that personal communication with known correspondents, news clipping services, mailing lists and nondelivery messages in general are not the things that are misclassified. What gets misclassified, is advertising, and the occasional cold call. Now, you look down at some of the poorer filters and you'll notice that right at the bottom of the list you actually have two filters that are very heavily touted on the Internet, to such an extent that I might even give them as an example of a disinformation campaign. They just misclassify everything -- personal email, bad email it's all the same.



University of Waterloo

False Positives by Genre

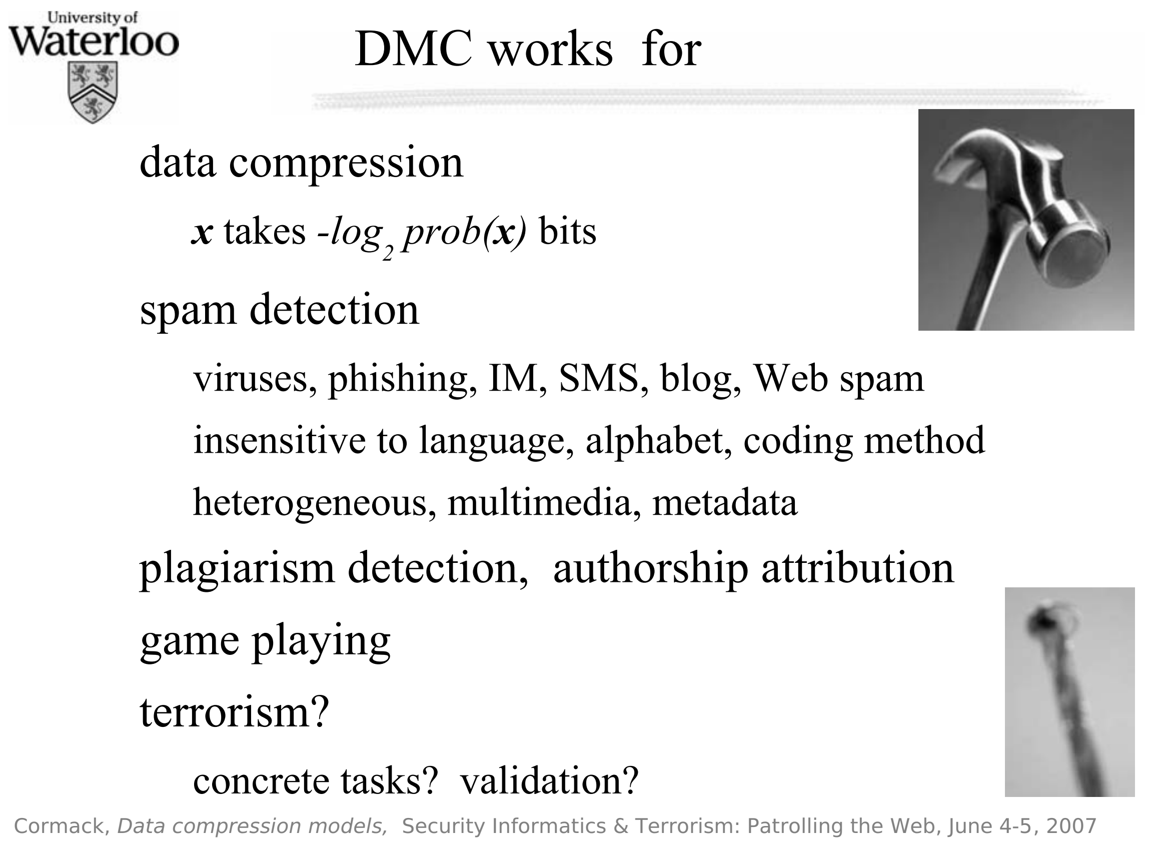
Filter	Advertising	Cold Call	Delivery	List	News	Personal	Transaction	Total
SA-Standard	4	2	0	0	0	0	0	6
SA-Super	1	0	0	1	1	0	3	6
Bogofilter	1	0	0	2	1	0	3	7
SA-Human	0	0	0	3	4	0	1	8
SA-Unsuper	5	0	0	1	0	1	3	10
SA-Bayes	1	0	0	4	1	1	8	15
SpamBayes	1	0	2	5	1	3	3	15
SA-Nolearn	1	0	4	0	3	9	0	17
SpamProbe	3	2	4	5	1	8	8	31
DSPAM	15	5	9	28	6	35	18	116
CRM114	7	15	13	78	10	135	37	295
Incoming Ham	0%	1%	17%	13%	14%	51%	4%	9038

Cormack, *Data compression models,* Security Informatics & Terrorism: Patrolling the Web, June 4-5, 2007

Figure 12. Qualitative "genre classification" [2]

All in all, we have to measure, we have to use measurements that are meaningful and our measurements have to be scientifically controllable. I can tell you that this works in a number of applications [Figure 13] because I can test it. If I can't test it what can I do? Try it out and see if it seems to work; hire an advertising company to promote it and then it doesn't matter whether it works because I have enough money anyway. What I can say for sure that DMC works for because I can test it as data compression. It works well for spam detection -- all the variants, viruses, phishing, different kinds of spam on the Web, and log spam. It's completely insensitive to the language of discourse, to the character encoding technique, it works great for multi-media, for multi-part MIME, and so on. It works pretty well for plagiarism and authorship detection, works pretty well for intrusion detection. I've already mentioned game playing but in this case I mean outright game playing.

Is it good for terrorism? I think it is, but terrorism is such an amorphous term I have no idea what you are talking about. So, if you have well-defined tasks -- better still, well-defined tasks and data – please talk to me. If you can send me your data -- which you will probably tell me is classified -- that's great, but if not, I can send you some software and you can scroll your data off and send me back one of those ROC curves and I'll be extremely happy.



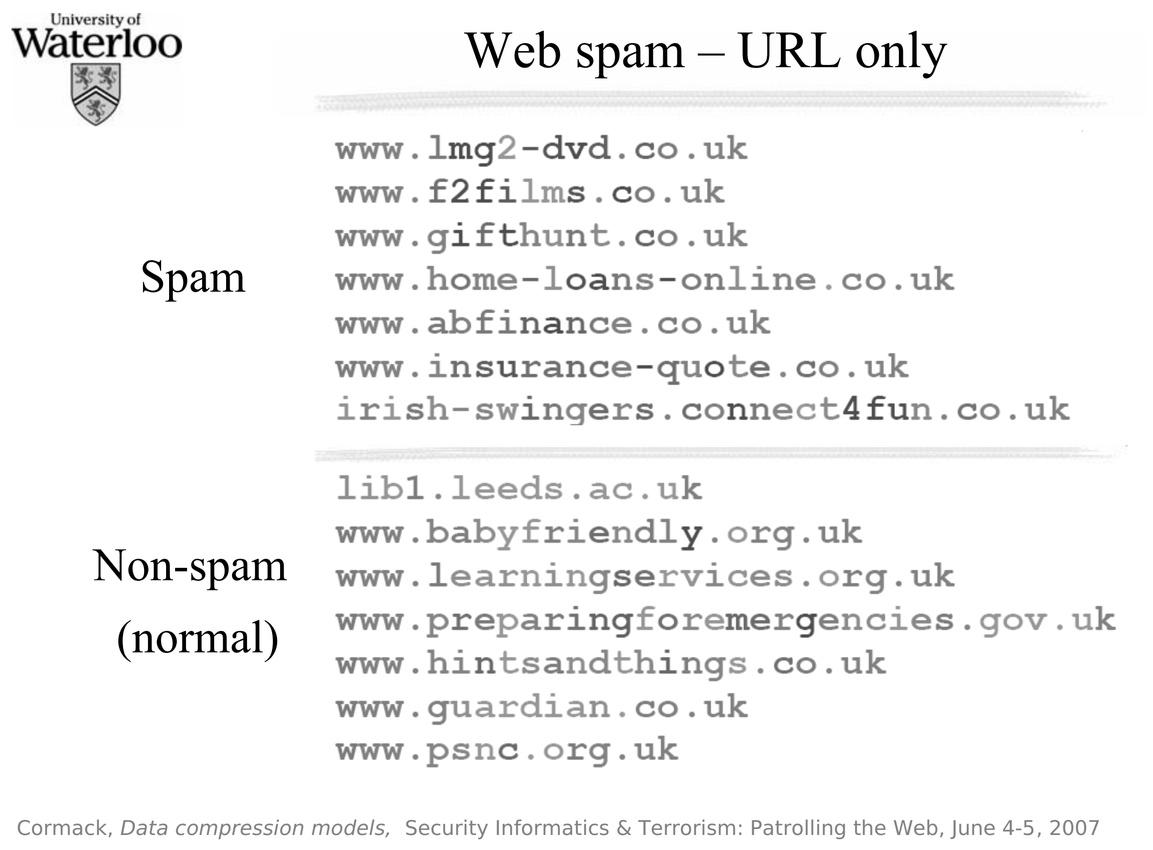
DMC works for

- data compression
  - x takes $-log_2\, prob(x)$ bits
- spam detection
  - viruses, phishing, IM, SMS, blog, Web spam
  - insensitive to language, alphabet, coding method
  - heterogeneous, multimedia, metadata
- plagiarism detection, authorship attribution
- game playing
- terrorism?
  - concrete tasks? validation?

Cormack, *Data compression models,* Security Informatics & Terrorism: Patrolling the Web, June 4-5, 2007

Figure 13. DMC applications

As far as "promoting DMC", I think I've actually said everything here. There are no explicit features, it handles heterogeneous data, it's adaptive, it learns on the job, there's none of this "find a training set and do this and then freeze the training set," so in terms of this concept drift and so on, it automatically captures new things that happen. I've already talked about visual output. It's extremely difficult for an adversary to defeat. And here's yet another example, and this is my final slide [Figure 14] which shows the application of DMC to web spam -- web pages that have no purpose other than to redirect you to other pages and to mislead search engines. This is DMC applied to the host name only, not even the URL, and you can see it actually does a pretty good job here as well.



Web spam – URL only

Spam	www.lmg2-dvd.co.uk
	www.f2films.co.uk
	www.gifthunt.co.uk
	www.home-loans-online.co.uk
	www.abfinance.co.uk
	www.insurance-quote.co.uk
	irish-swingers.connect4fun.co.uk
Non-spam (normal)	lib1.leeds.ac.uk
	www.babyfriendly.org.uk
	www.learningservices.org.uk
	www.preparingforemergencies.gov.uk
	www.hintsandthings.co.uk
	www.guardian.co.uk
	www.psnc.org.uk

Cormack, *Data compression models,* Security Informatics & Terrorism: Patrolling the Web, June 4-5, 2007

Figure 14. Web spam hostnames

Thank you.

References

[1] Andrej Bratko, Gordon V. Cormack, Bogdan Filipič, Thomas R. Lynam and Blaž Zupan, Spam filtering using statistical data compression models, *J. Machine Learning Research* 7 (Dec. 2006), 2673--2698.

[2] Gordon V. Cormack and Thomas R. Lynam, On-line supervised spam filtering, *ACM Trans. Information Systems* 25 (July 2007).