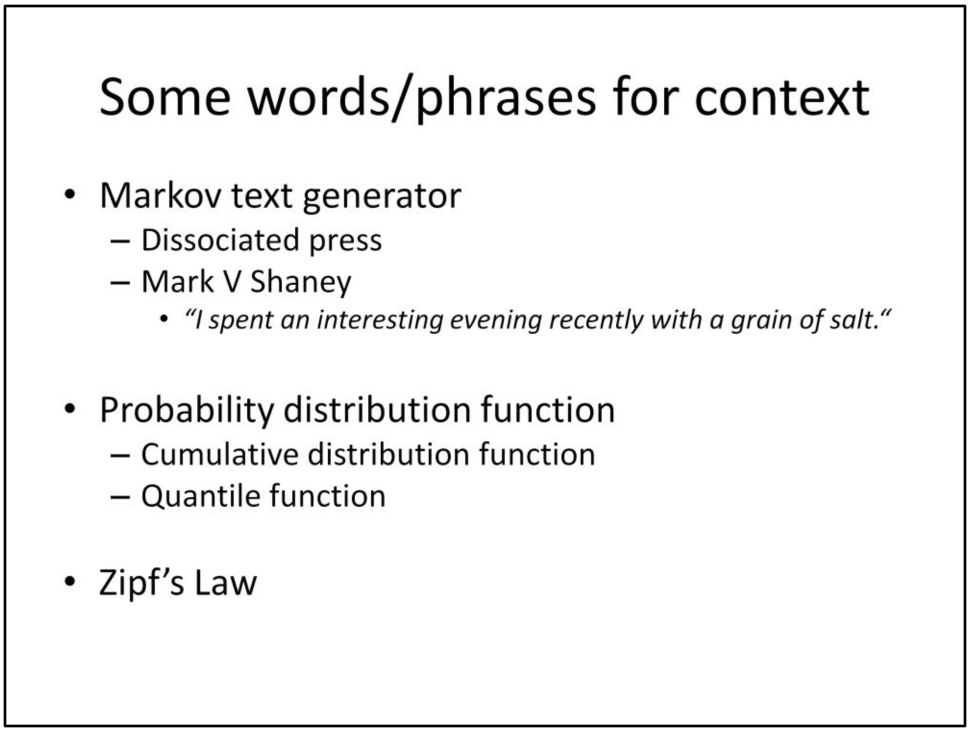
“All Your Secret Data Makes a Great Nigerian Prince”

Encoding data for quantitative and qualitative properties through greedy algorithms for no reason at all

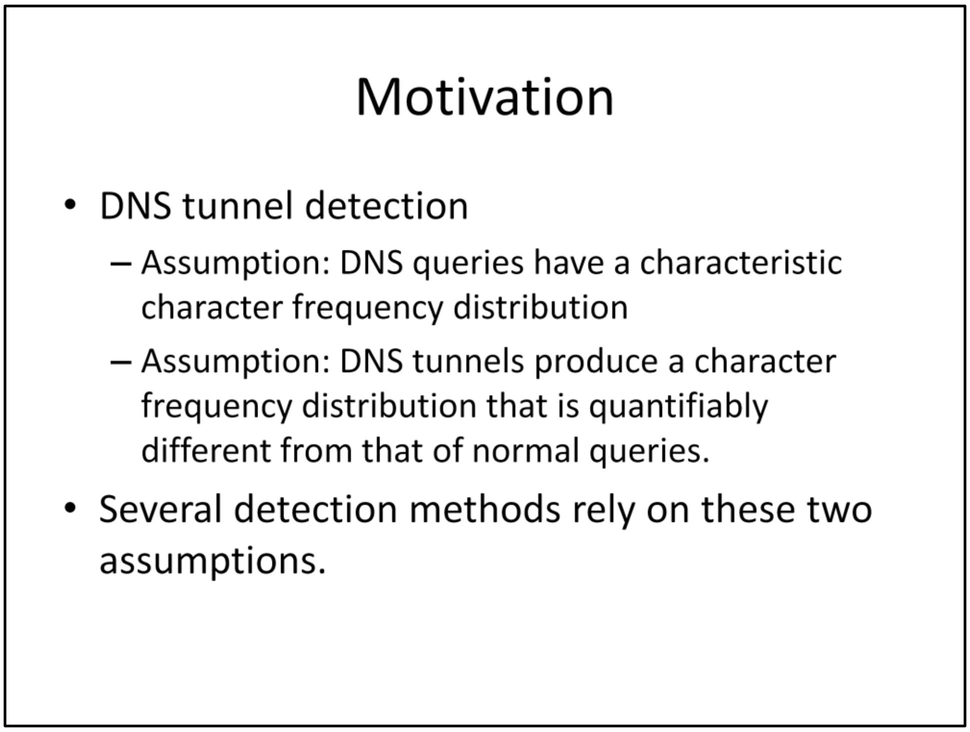


Some words/phrases for context

- Markov text generator
  - Dissociated press
  - Mark V Shaney
    - *"I spent an interesting evening recently with a grain of salt."*
- Probability distribution function
  - Cumulative distribution function
  - Quantile function
- Zipf's Law

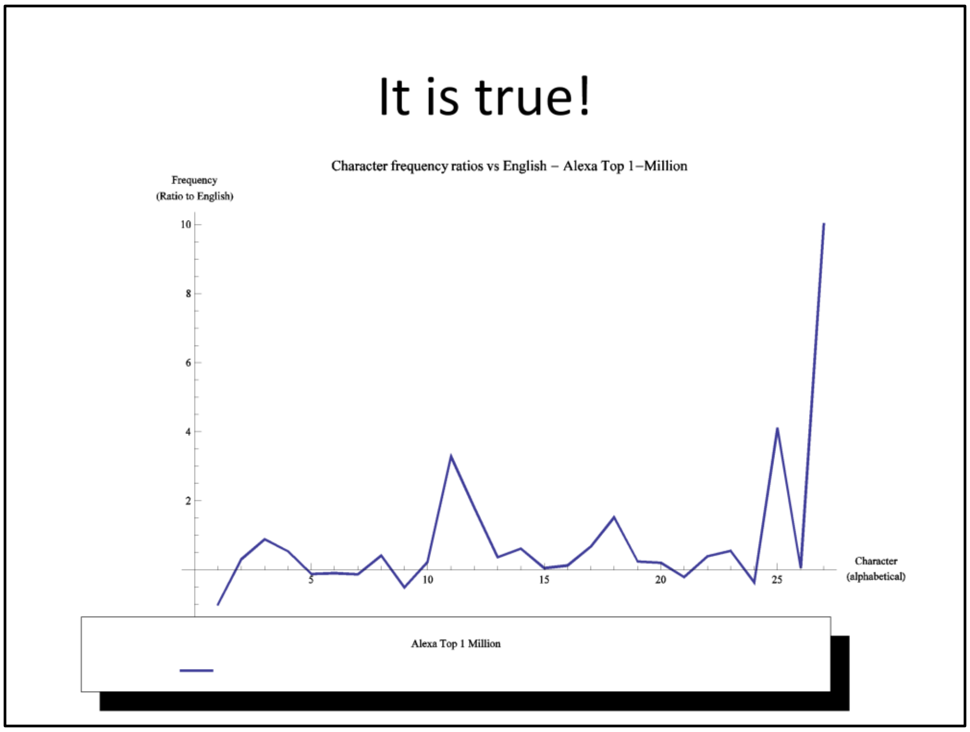
-Quantile function
    -Inverse of CDF
    -Input uniform distribution
    -get out some other ditribution defined by the CDF you inverted

-Zipf's law
    -The frequency of a word is
    -inversely proportional to its
    -rank in the frequency table.



Motivation

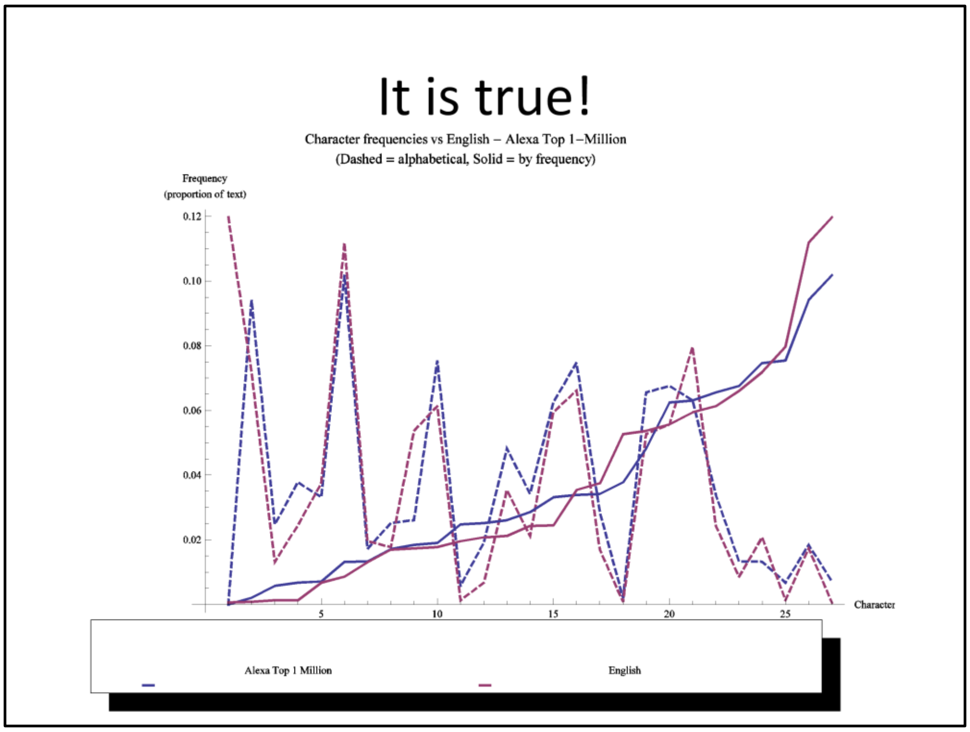
- DNS tunnel detection
  – Assumption: DNS queries have a characteristic character frequency distribution
  – Assumption: DNS tunnels produce a character frequency distribution that is quantifiably different from that of normal queries.
- Several detection methods rely on these two assumptions.



It is true!

-Actually a pretty robust distribution

-Shuffle the list, take a few hundred and the distribution starts to appear.

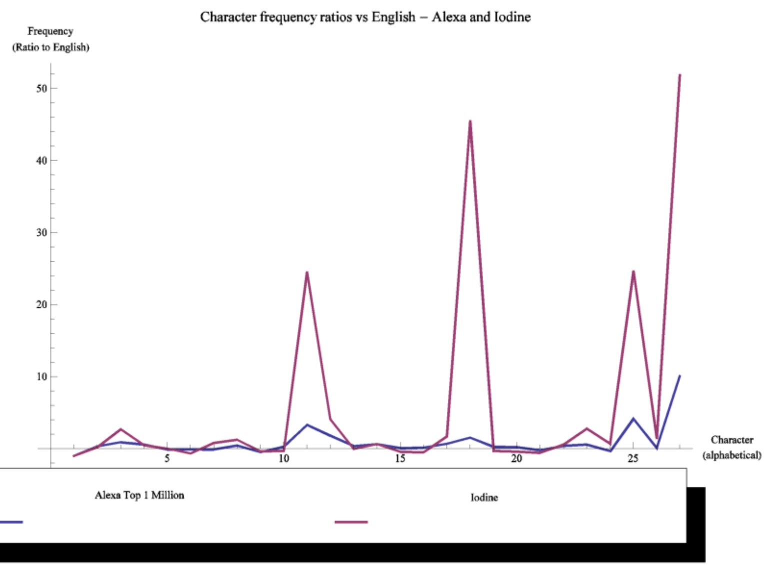


It is true!

Note that it follows english,
but doesn't actually mirror it.

There are marked differences.

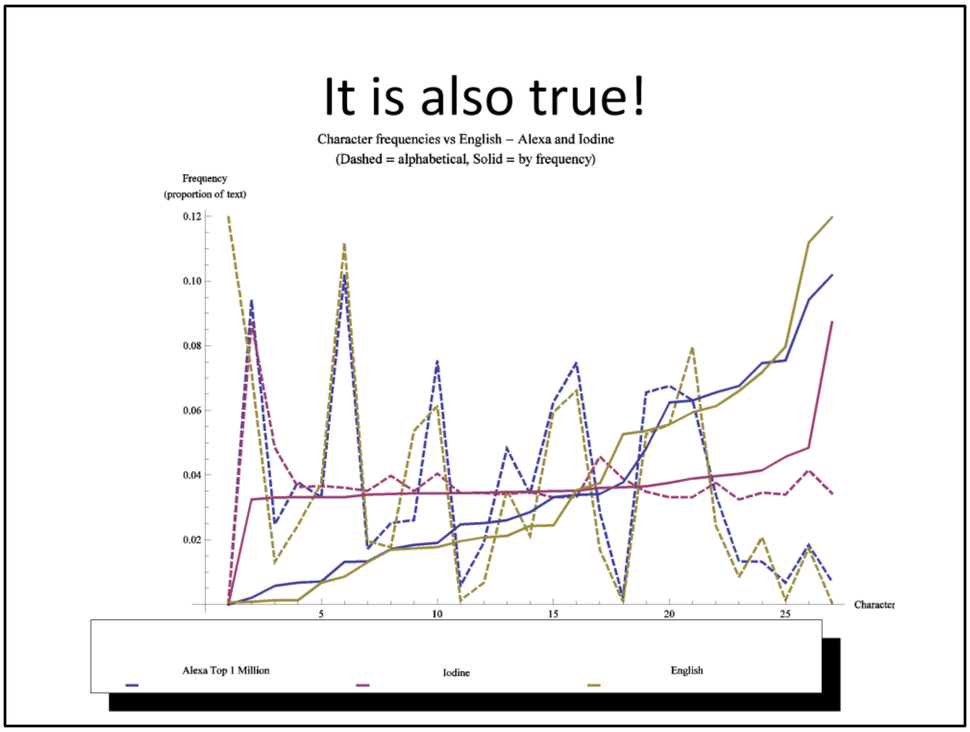
It is also true!



I chose iodine

This plot is deceptive

Apparently only a few deviants

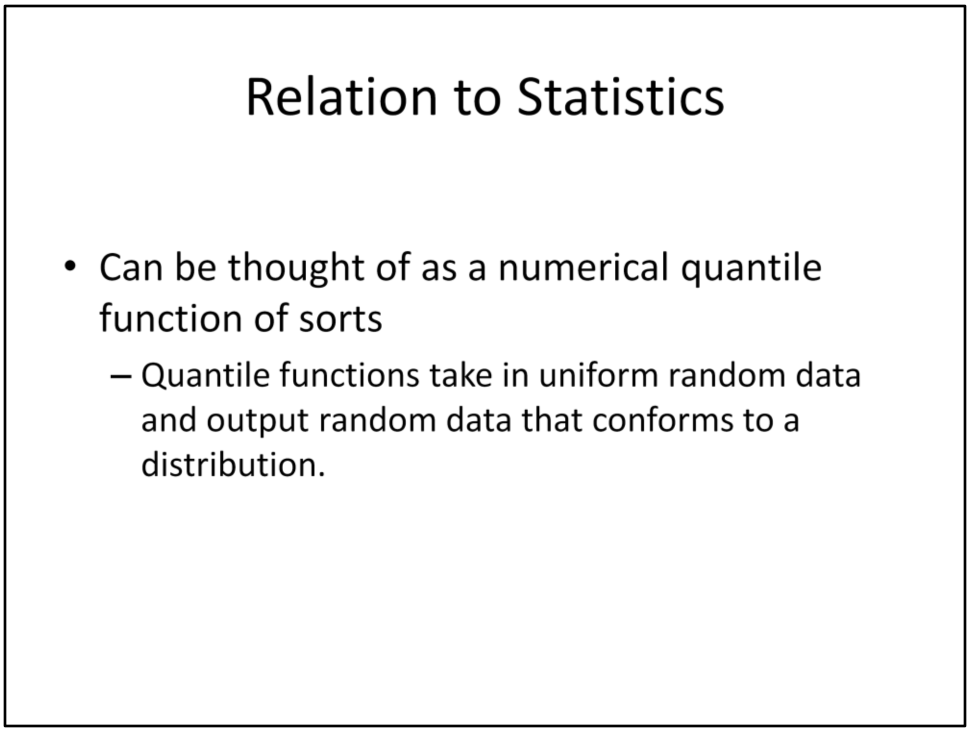


It is also true!

Easily discerned using a number
of different classifiers or statistics.

... But what if it weren’t?

- Then these detection methods would fail miserably.
- Is it possible to manipulate arbitrary data into a form that possesses certain statistical properties?

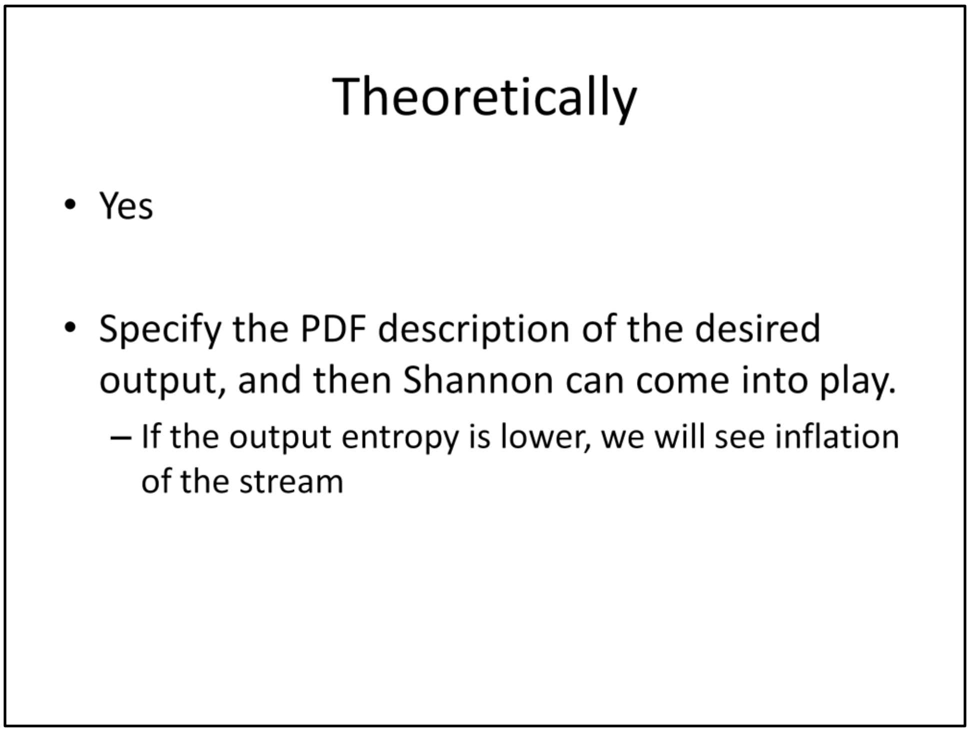


Relation to Statistics

- Can be thought of as a numerical quantile function of sorts
  - Quantile functions take in uniform random data and output random data that conforms to a distribution.

CDf bijective from domain to [0,1]
So invertible

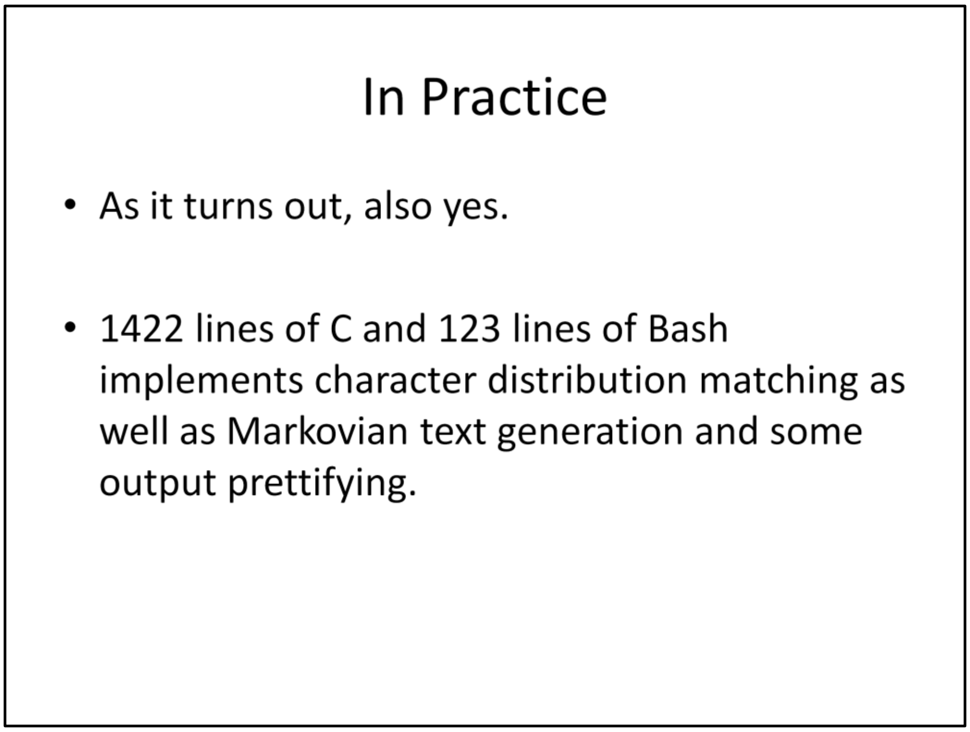
But not normally computable.



Theoretically

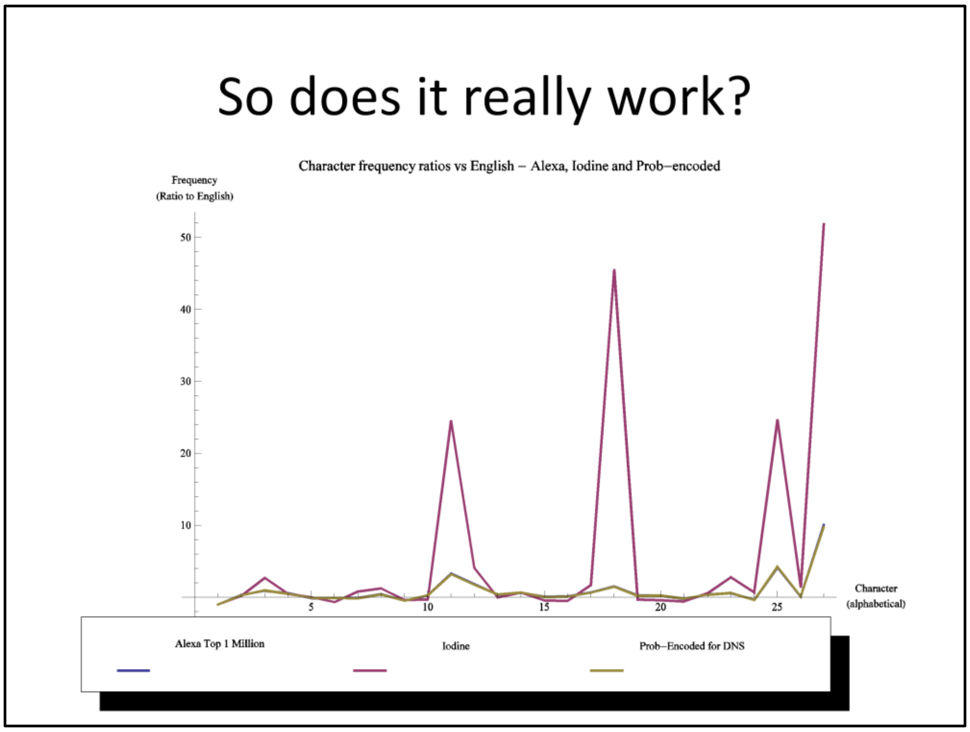
- Yes
- Specify the PDF description of the desired output, and then Shannon can come into play.
  - If the output entropy is lower, we will see inflation of the stream

This is roughly equivalent to transmitting something over a rate-limited channel.



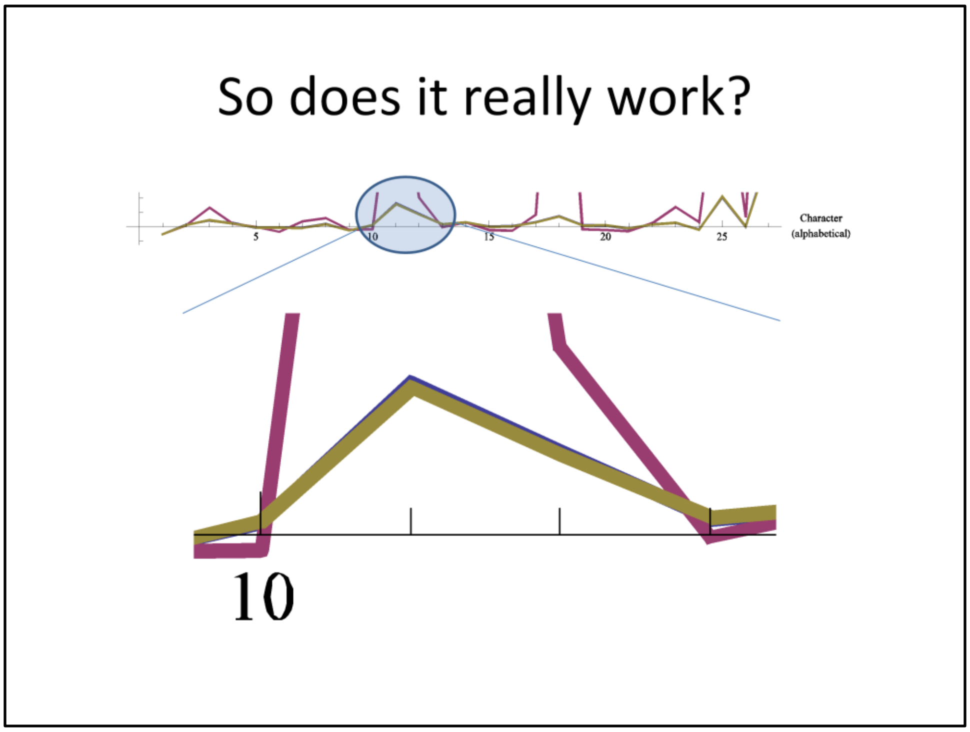
In Practice

- As it turns out, also yes.
- 1422 lines of C and 123 lines of Bash implements character distribution matching as well as Markovian text generation and some output prettifying.



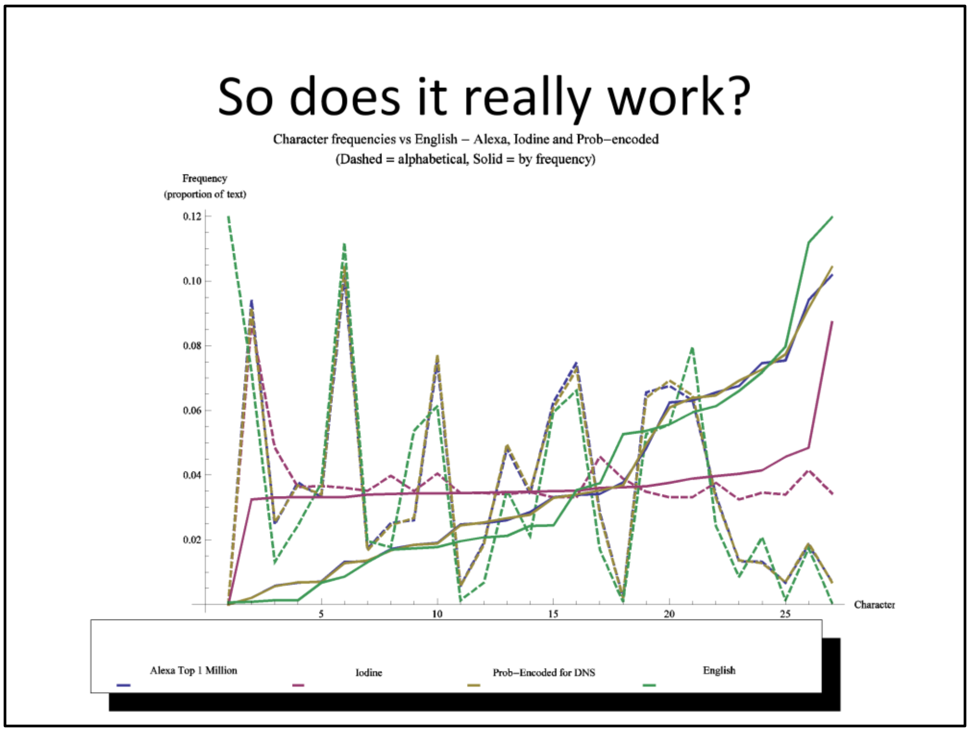
So does it really work?

There are actually three lines.



So does it really work?

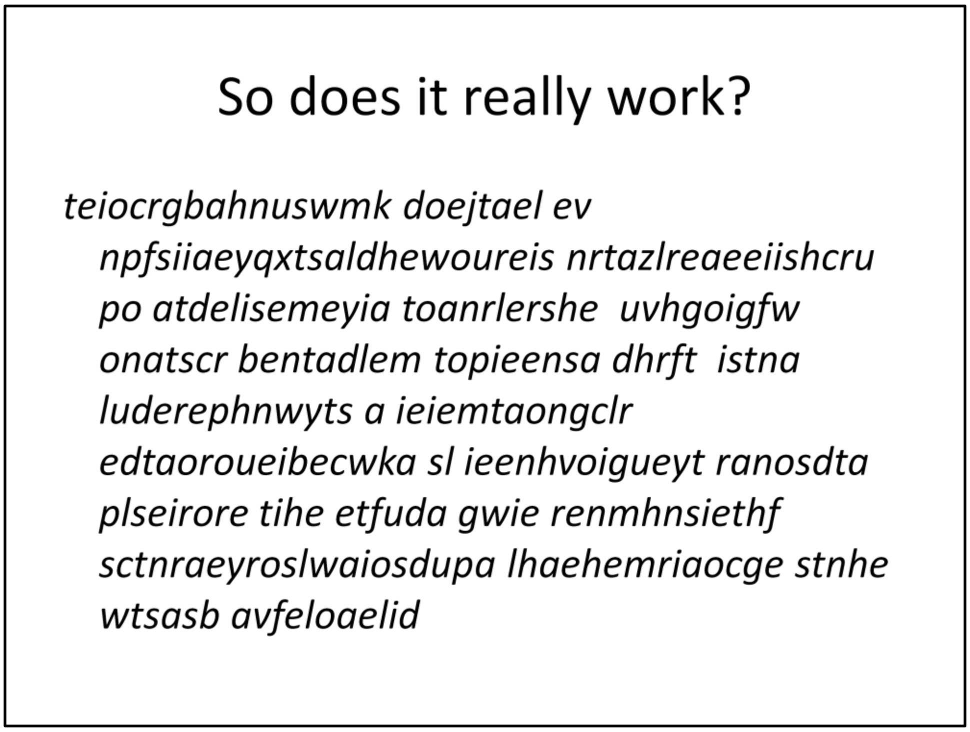
The differences are so small they
are barely visible in the plots.



So does it really work?

Overall match is very good
Some variation

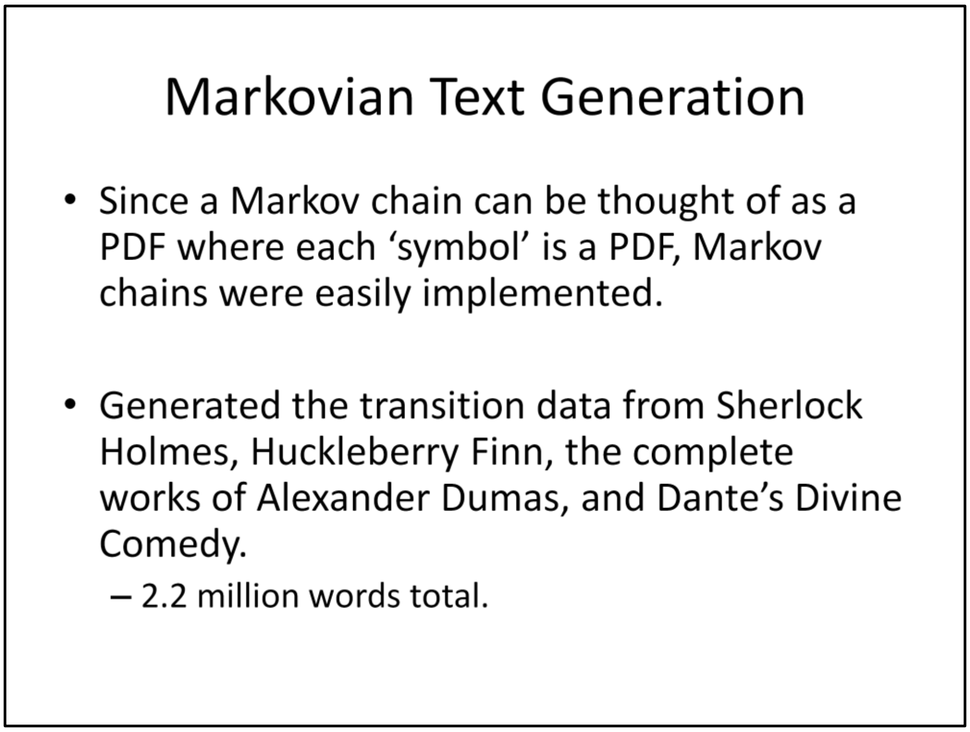
Statistically difficult to tell them apart



So does it really work?

teiocrgbahnuswmk doejtael ev npfsiiaeyqxtsaldhewoureis nrtazlreaeeiishcru po atdelisemeyia toanrlershe uvhgoigfw onatscr bentadlem topieensa dhrft istna luderephnwyts a ieiemtaongclr edtaoroueibecwka sl ieenhvoigueyt ranosdta plseirore tihe etfuda gwie renmhnsiethf sctnraeyroslwaiosdupa lhaehemriaocge stnhe wtsasb avfeloaelid

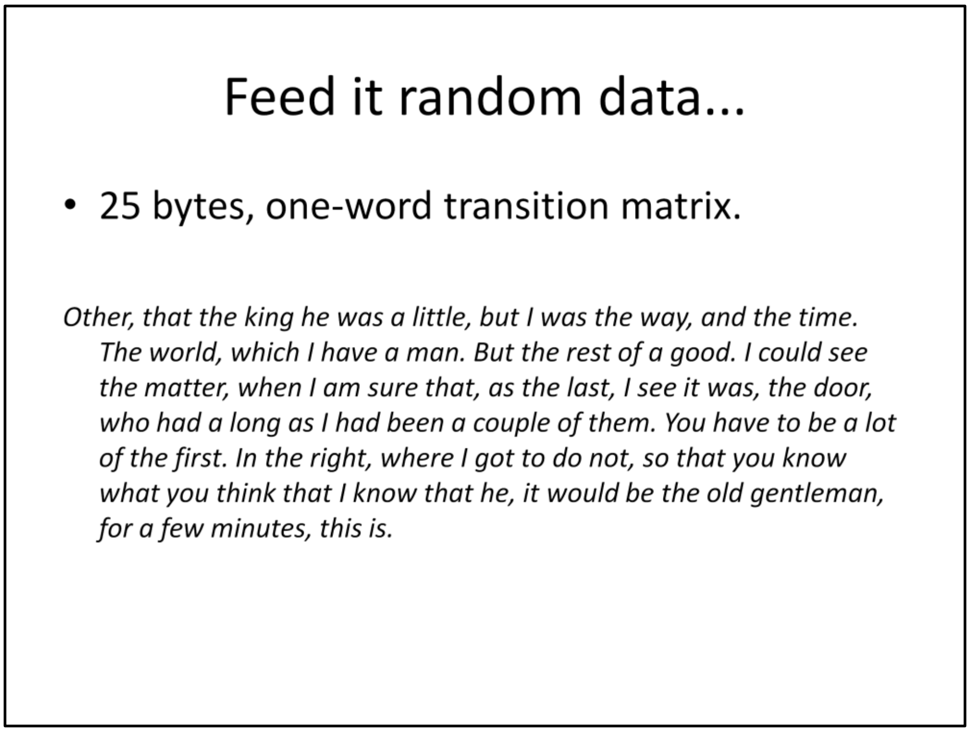
Until you actually look at its output.



Markovian Text Generation

- Since a Markov chain can be thought of as a PDF where each ‘symbol’ is a PDF, Markov chains were easily implemented.
- Generated the transition data from Sherlock Holmes, Huckleberry Finn, the complete works of Alexander Dumas, and Dante’s Divine Comedy.
  - 2.2 million words total.

Markov text generation can work
really well given appropriate input material.

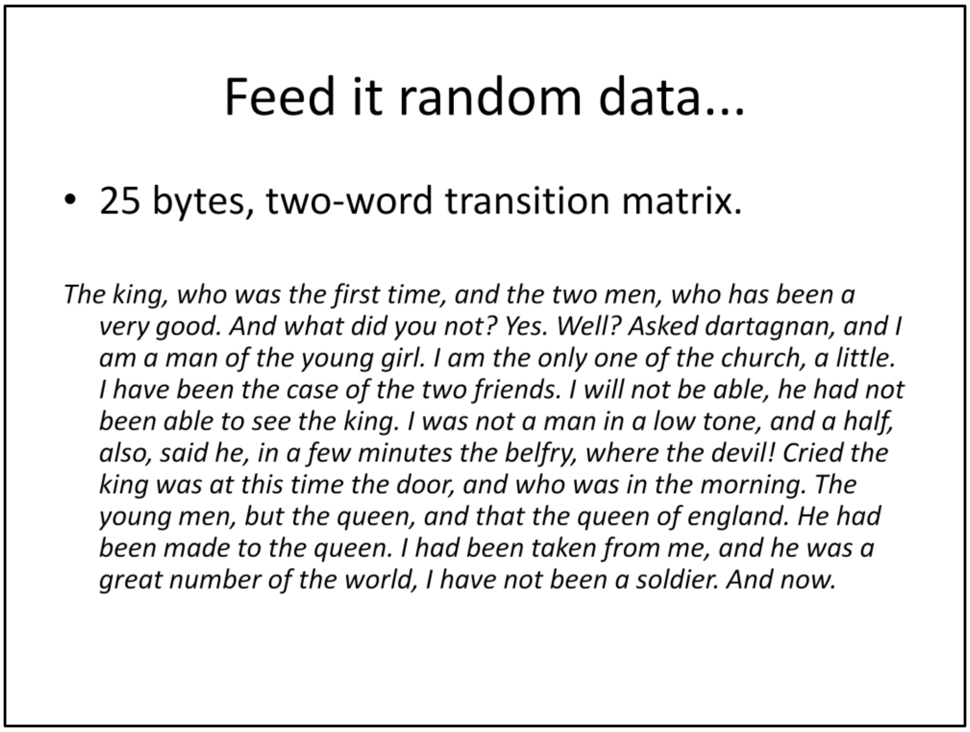


Feed it random data...

- 25 bytes, one-word transition matrix.

Other, that the king he was a little, but I was the way, and the time. The world, which I have a man. But the rest of a good. I could see the matter, when I am sure that, as the last, I see it was, the door, who had a long as I had been a couple of them. You have to be a lot of the first. In the right, where I got to do not, so that you know what you think that I know that he, it would be the old gentleman, for a few minutes, this is.

Pretty broken, and sentences just feel random.

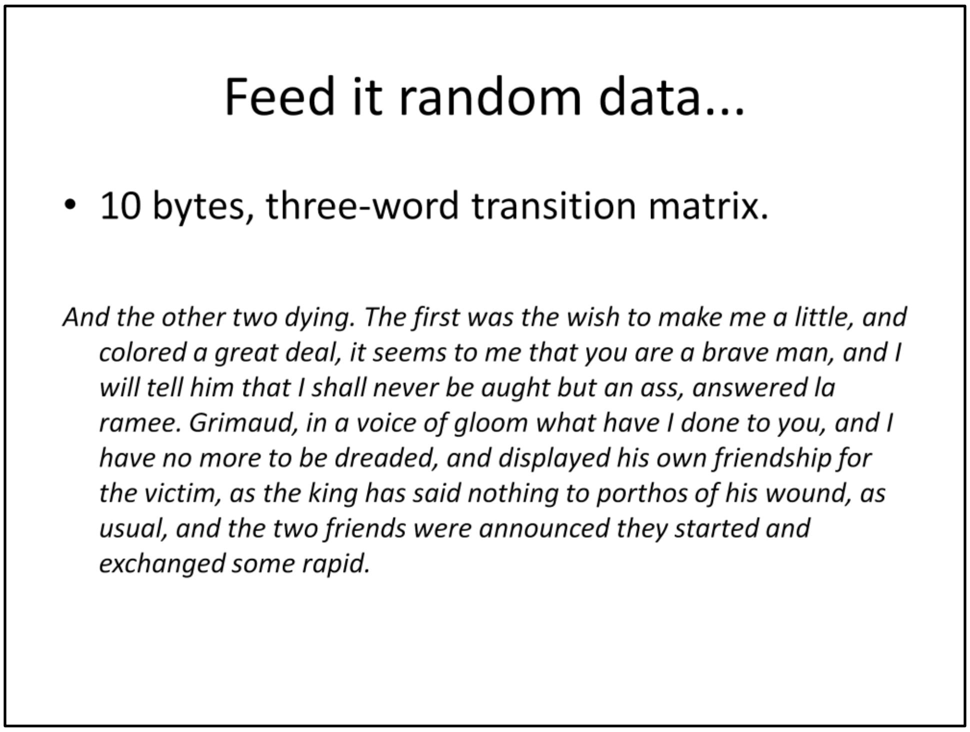


Feed it random data...

- 25 bytes, two-word transition matrix.

The king, who was the first time, and the two men, who has been a very good. And what did you not? Yes. Well? Asked dartagnan, and I am a man of the young girl. I am the only one of the church, a little. I have been the case of the two friends. I will not be able, he had not been able to see the king. I was not a man in a low tone, and a half, also, said he, in a few minutes the belfry, where the devil! Cried the king was at this time the door, and who was in the morning. The young men, but the queen, and that the queen of england. He had been made to the queen. I had been taken from me, and he was a great number of the world, I have not been a soldier. And now.

Getting better...



Feed it random data...

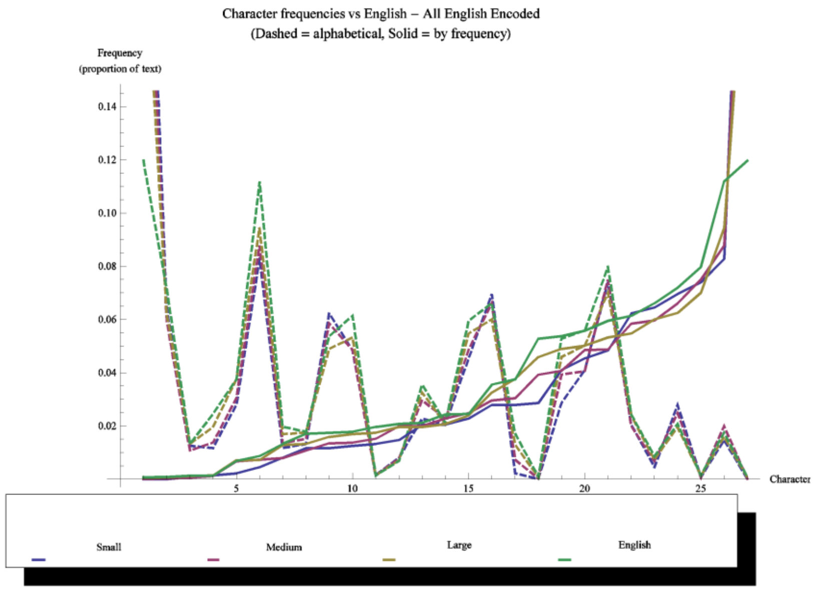
- 10 bytes, three-word transition matrix.

And the other two dying. The first was the wish to make me a little, and colored a great deal, it seems to me that you are a brave man, and I will tell him that I shall never be aught but an ass, answered la ramee. Grimaud, in a voice of gloom what have I done to you, and I have no more to be dreaded, and displayed his own friendship for the victim, as the king has said nothing to porthos of his wound, as usual, and the two friends were announced they started and exchanged some rapid.

Maybe someone wrote feelings
poetry after a few too many beers?

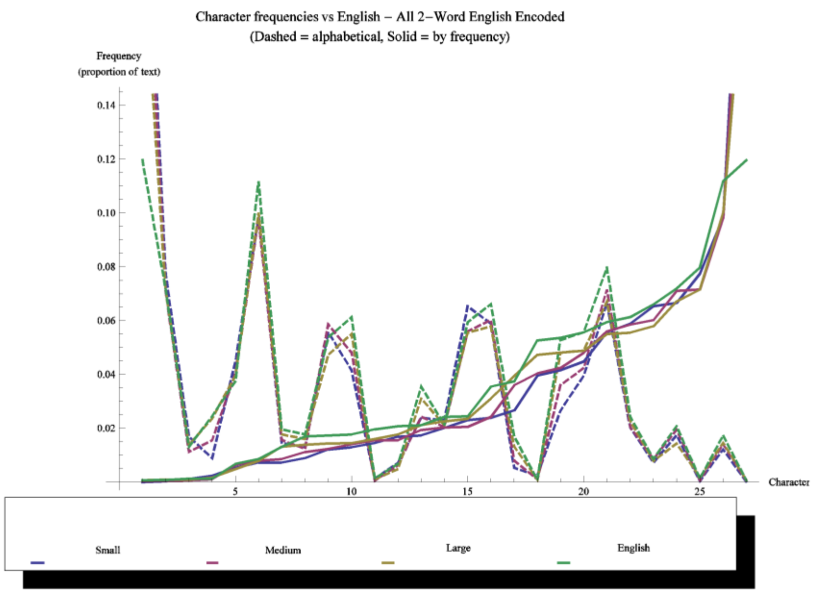
Getting better.

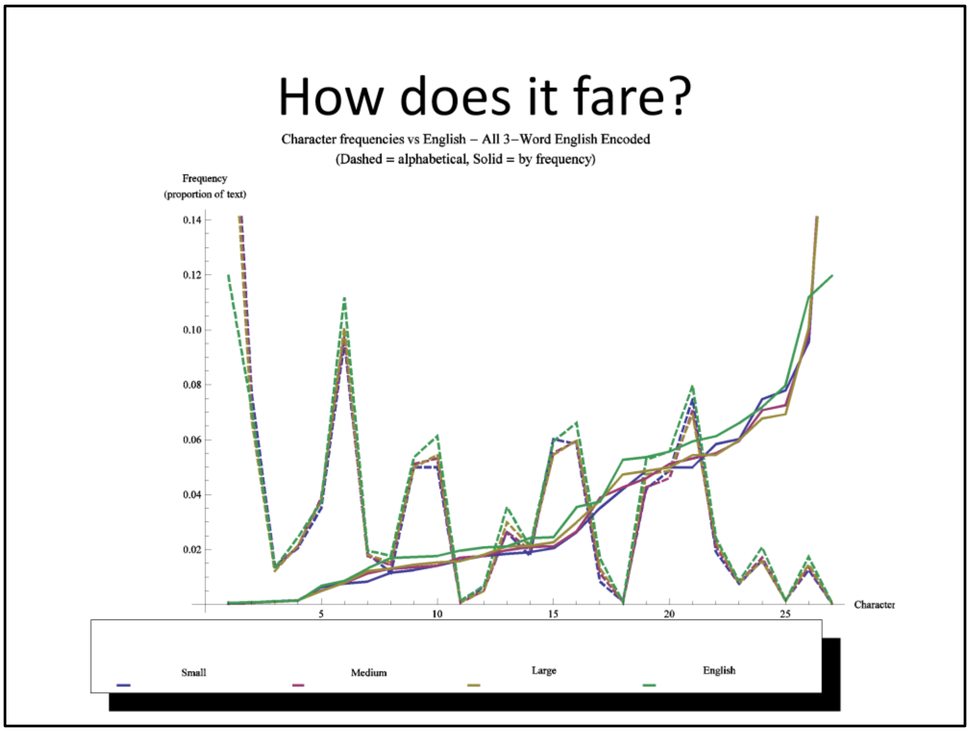
How does it fare?



Small, medium, and large inputs.
100, 10000 and 100000 bytes respectively.

How does it fare?

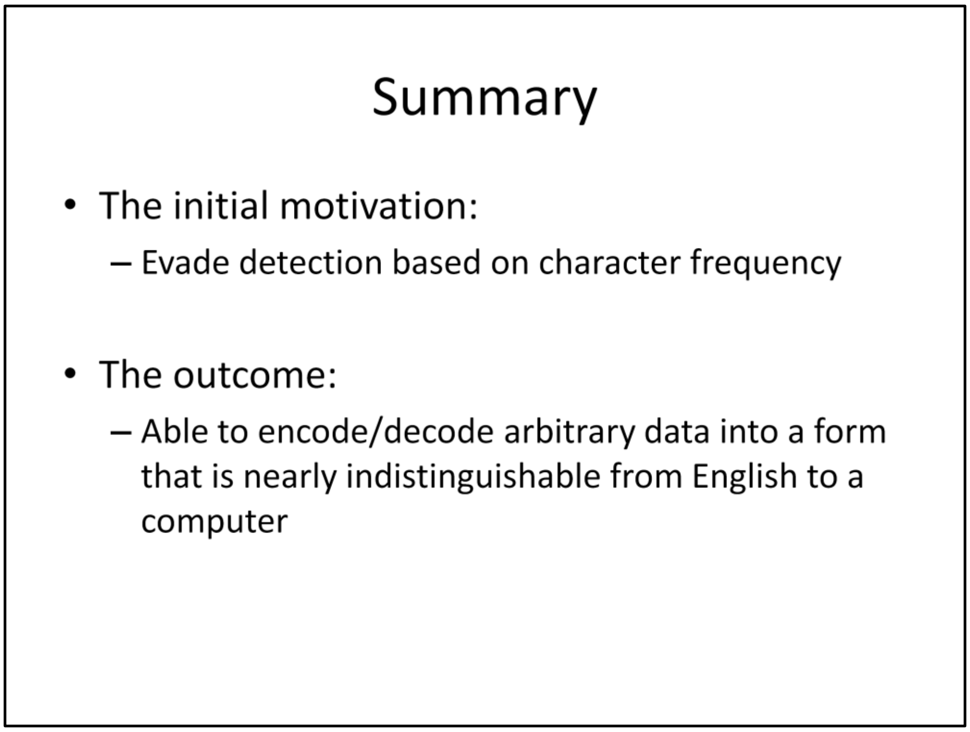




How does it fare?

You can see it tighten up.

The large corpus follows the trends of english very well.



Summary

- The initial motivation:
  - Evade detection based on character frequency
- The outcome:
  - Able to encode/decode arbitrary data into a form that is nearly indistinguishable from English to a computer

Everything we've done so far we can undo!

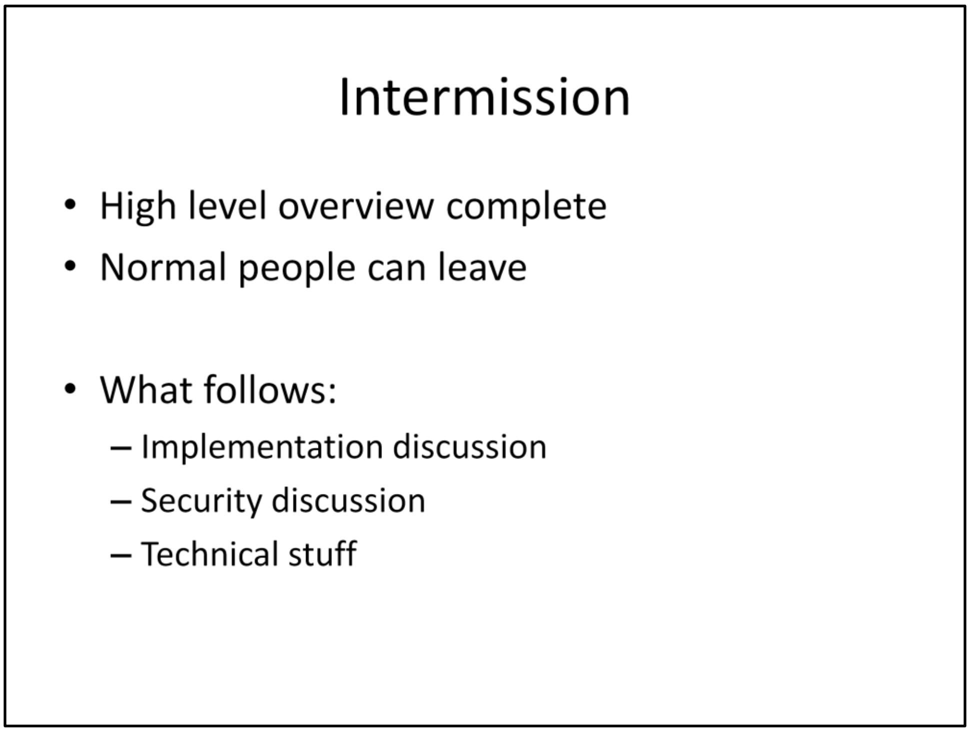
This is a completely reversible encoding!

Why?

- What practical use does this have?

Possibility: Covert Instruction Spam

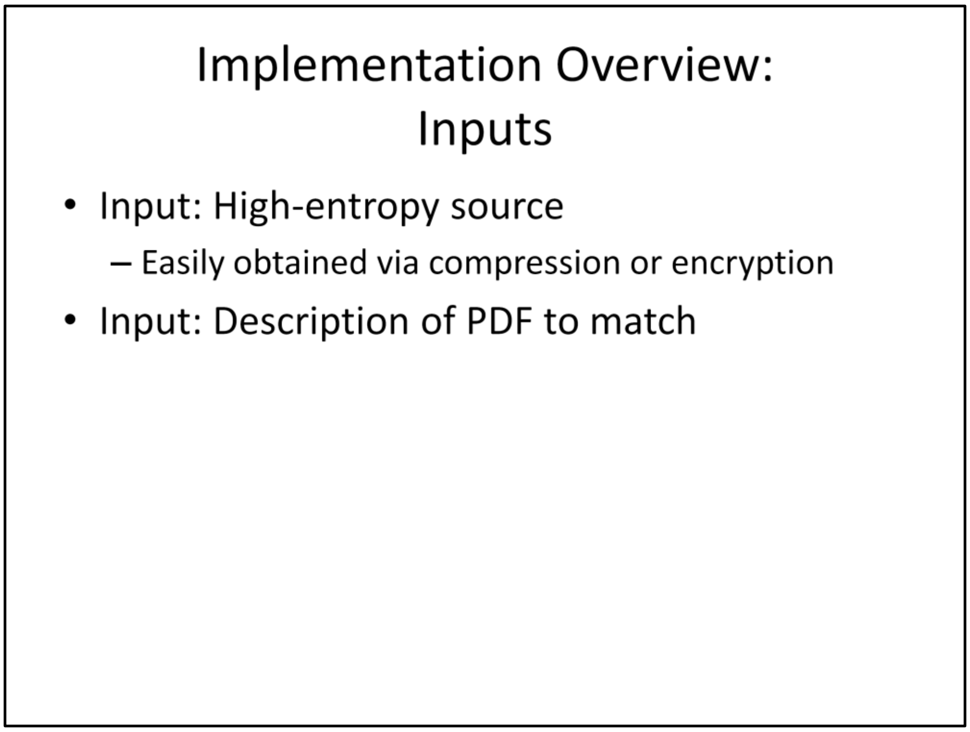
- How could you send secret data to an operative in the field, without anyone knowing you sent anything to them?
  - Encode the covert data as spammy English, and send it to millions of people.
  - How would you identify the intended recipient?



Intermission

- High level overview complete
- Normal people can leave
- What follows:
  - Implementation discussion
  - Security discussion
  - Technical stuff

Questions?



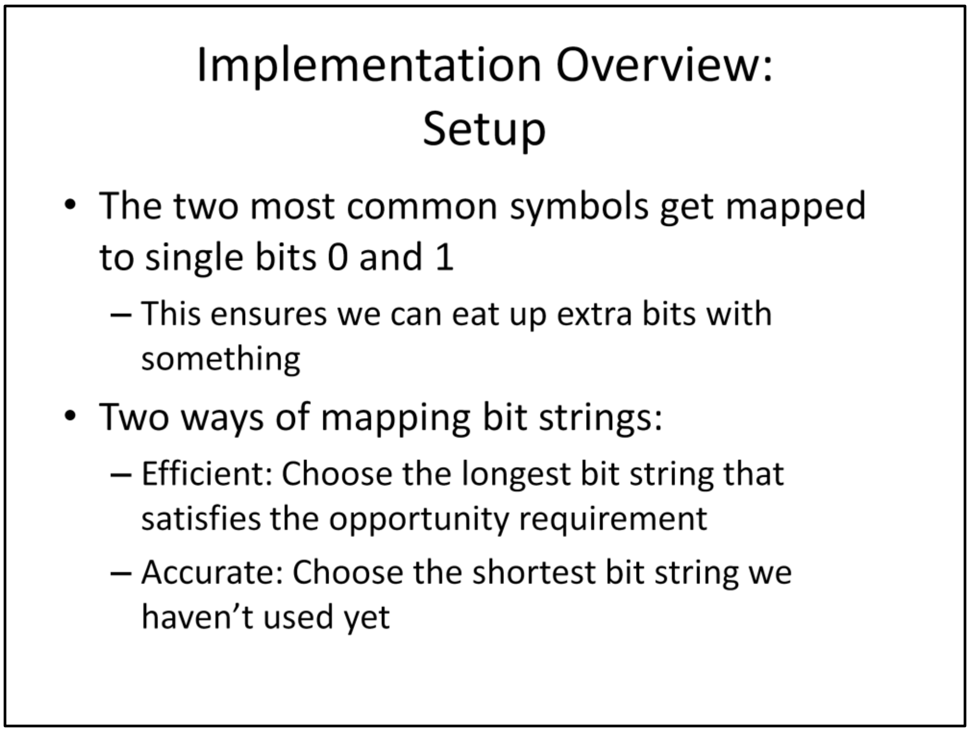
Implementation Overview: Inputs

- Input: High-entropy source
  - Easily obtained via compression or encryption
- Input: Description of PDF to match

The reason for a high entropy source will be clear shortly

Implementation Overview: Setup

- Sort PDF from most to least probable symbol
- Map between symbols and bit strings
  - Prefix-free mappings are not a requirement here (you'll see why later)
  - Each symbol gets a bit string that is at least as likely in a high-entropy input as the symbol should be in the output
    - This ensures that we get enough opportunities to choose this as the output symbol



Implementation Overview: Setup

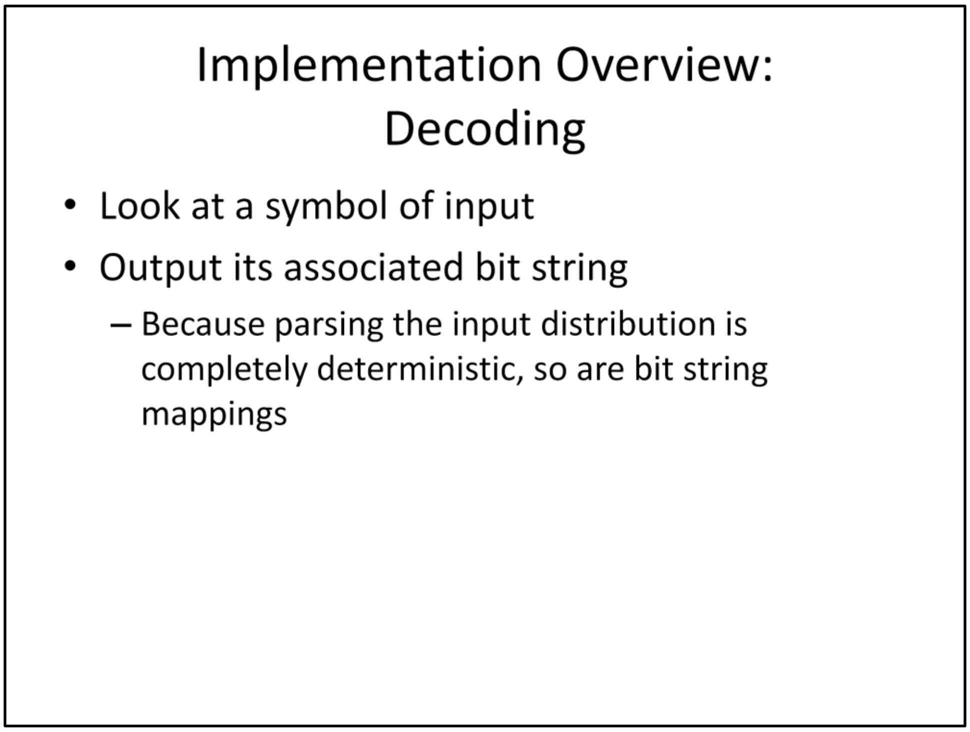
- The two most common symbols get mapped to single bits 0 and 1
  - This ensures we can eat up extra bits with something
- Two ways of mapping bit strings:
  - Efficient: Choose the longest bit string that satisfies the opportunity requirement
  - Accurate: Choose the shortest bit string we haven’t used yet

The implication here is that the
most probably output symbol cannot
have probability greater than 50%,
or we won’t be able to give it enough
opportunities.

Efficient mapping has a considerable
impact on output distribution for a reasonable
savings in output size (20%-ish).

Implementation Overview: Encoding (Greedy Algorithm)

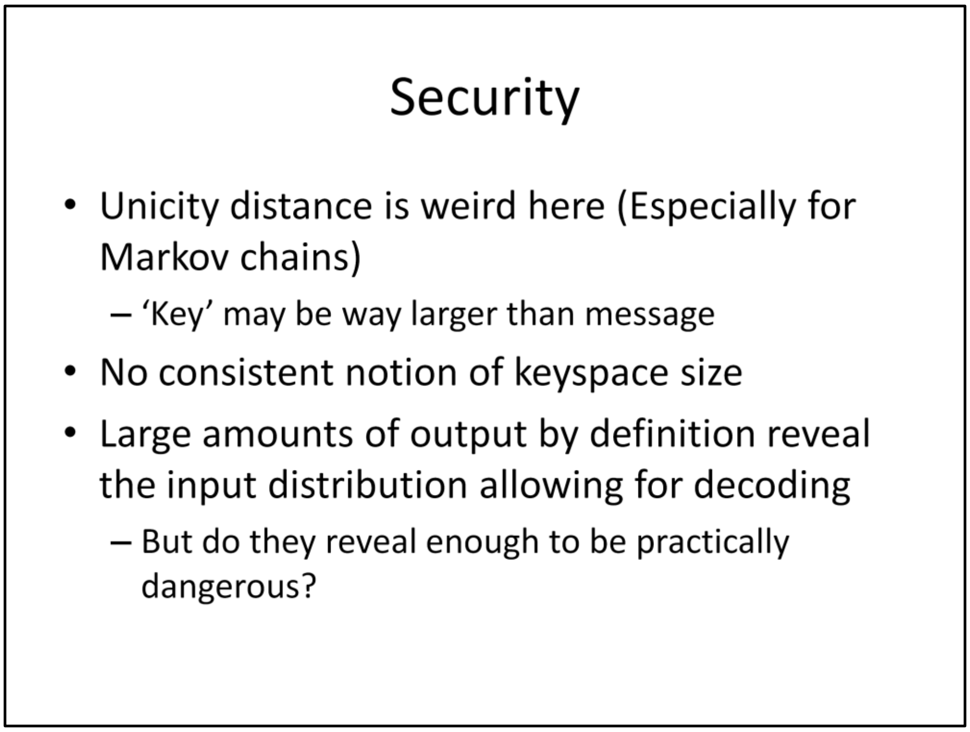
- Look at the first *N* bits of input, where *N* is the maximum length of a bit string we mapped to a symbol
- Find the symbols that could match (a portion of) the bits
  - Choose the one that is farthest from reaching its quota so far (greedy portion). Resolve ties arbitrarily.



Implementation Overview: Decoding

- Look at a symbol of input
- Output its associated bit string
  - Because parsing the input distribution is completely deterministic, so are bit string mappings

Decoding is really straight forward.



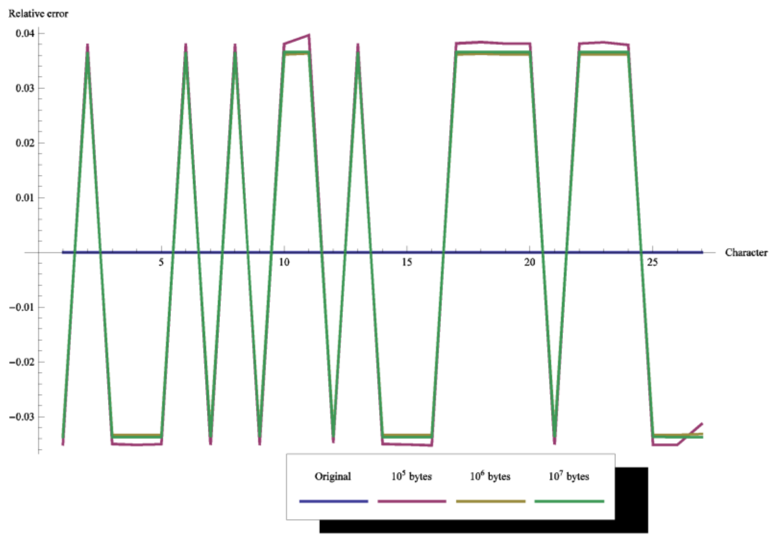
Security

- Unicity distance is weird here (Especially for Markov chains)
  - 'Key' may be way larger than message
- No consistent notion of keyspace size
- Large amounts of output by definition reveal the input distribution allowing for decoding
  - But do they reveal enough to be practically dangerous?

Unicity distance is how much encoded content you need to see before a spurious key will show itself as spurious.

What is the 'size' of a Markov chain distribution? Is it bounded?

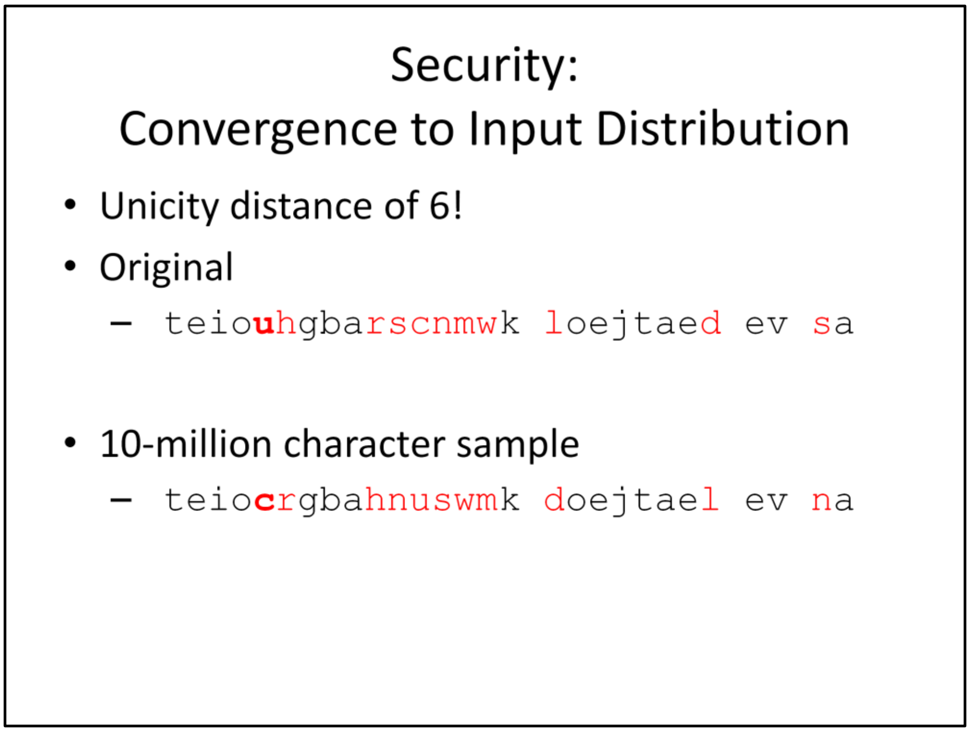
Security:
Convergence to Input Distribution



Data came from /dev/urandom

Note the systemic differences.

They are ALL off by three to four percent.



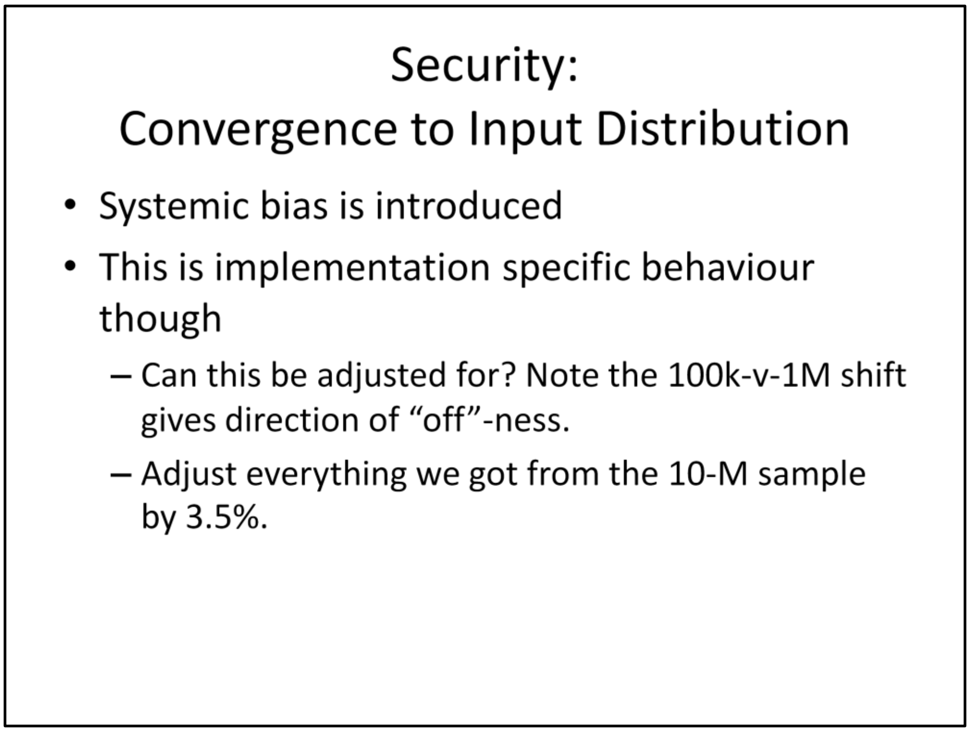
Security: Convergence to Input Distribution

- Unicity distance of 6!
- Original
  - `teio`**`u`**`hgbarscnmwk loejtaed ev sa`
- 10-million character sample
  - `teio`**`c`**`rgbahnuswmk doejtael ev na`

Took the original distribution specification,
and encoded a few bytes.

Then encoded ten million bytes of /dev/urandom,
converted that into a distribution specification,
and encoded the same few bytes as with the original.

Not everything is wrong, but enough and soon enough.

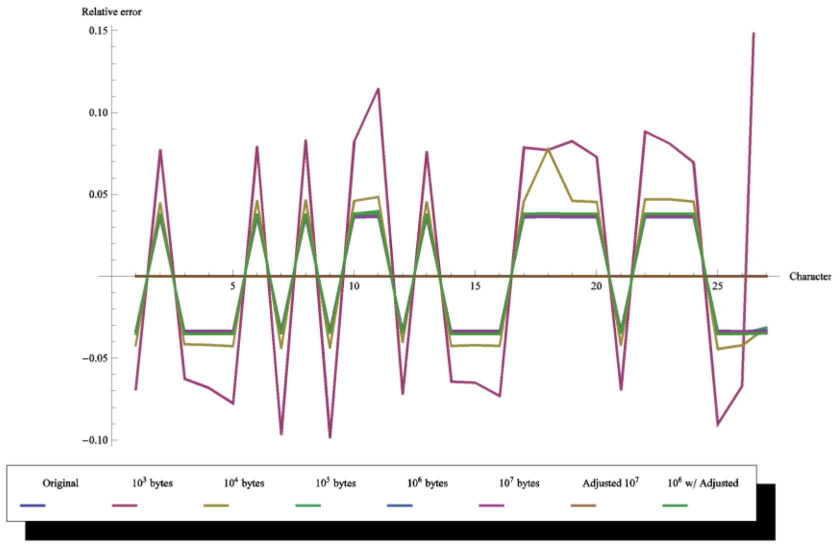


Security: Convergence to Input Distribution

- Systemic bias is introduced
- This is implementation specific behaviour though
  - Can this be adjusted for? Note the 100k-v-1M shift gives direction of "off"-ness.
  - Adjust everything we got from the 10-M sample by 3.5%.

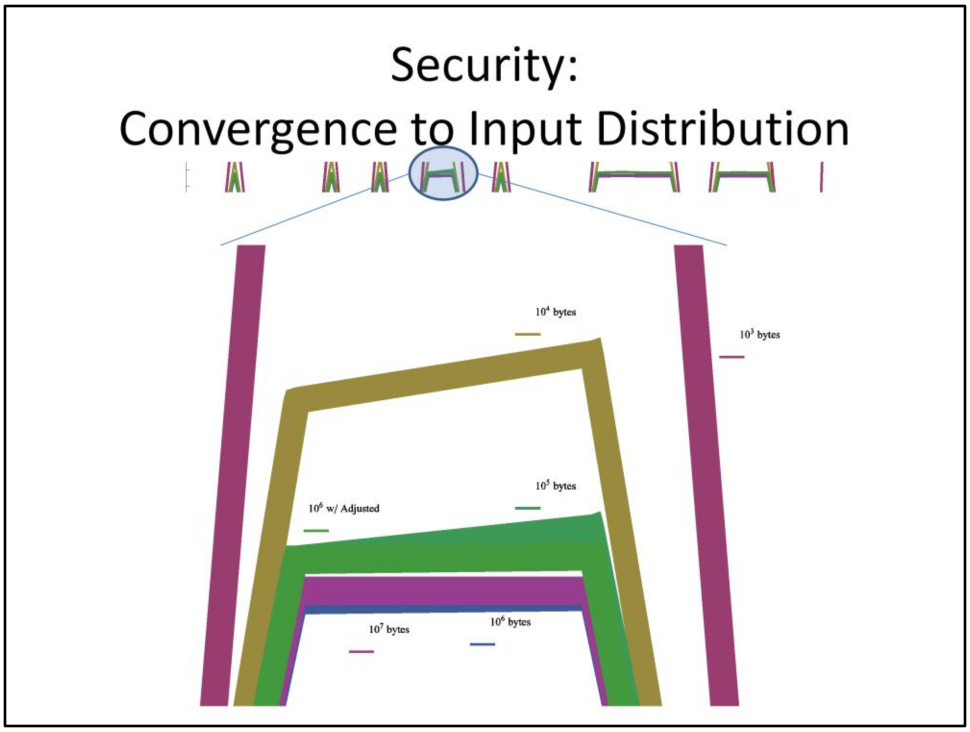
Different implementations will introduce different biases.

Security: Convergence to Input Distribution



I added in 10^3 and 10^4 byte samples
to demonstrate that shorter amounts of
text demonstrate more variability in error.

It is the uniformity of error that is exploitable.

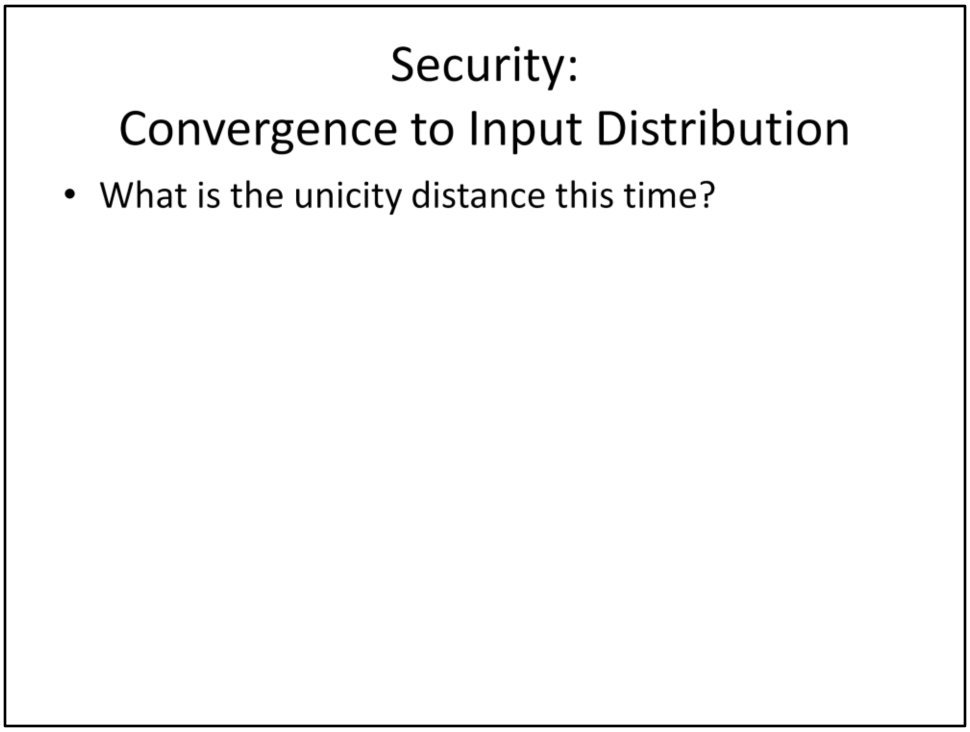

Security:
Convergence to Input Distribution

A closer view of one section.

We can see how close everything is.

Note that even though it is a sample
of 10^6 bytes of input, it doesn’t match
as well as the ‘real’ 10^6 sample.

The 10^6 adjusted plot is covering the
10^5 byte plot, but you can just barely
make out a sliver of the 10^5 byte plot behind it.



Security: Convergence to Input Distribution

- What is the unicity distance this time?

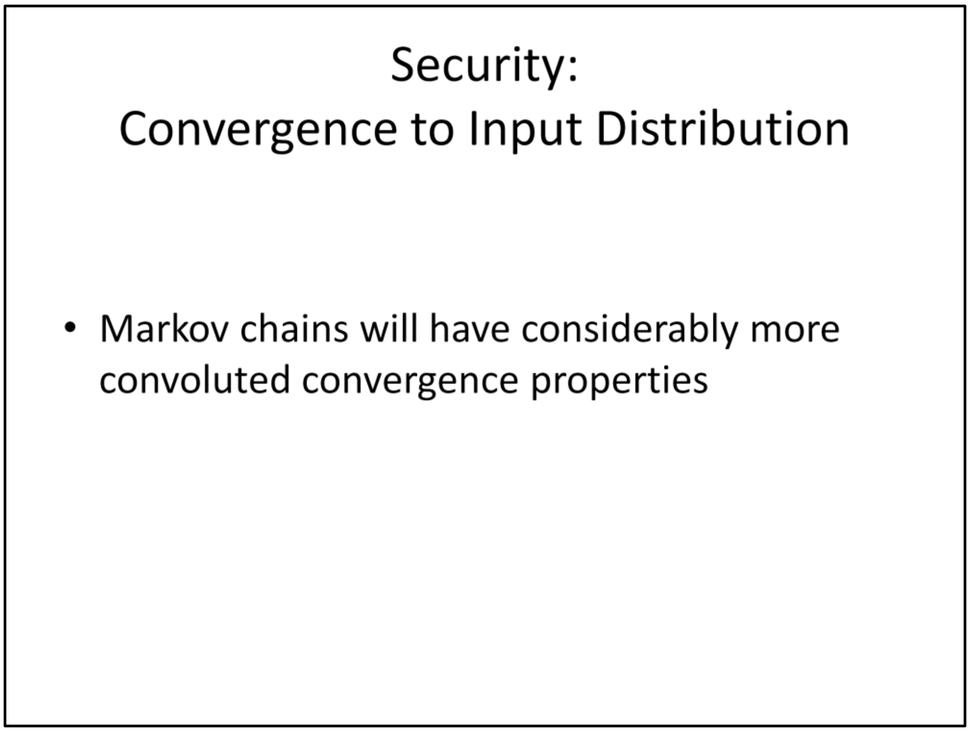
Guesses?

Security:
Convergence to Input Distribution

- What is the unicity distance this time?
- 9899 characters of output!

- Original:
  - …`at euieno amtwfpvldoshs iretcra enogorsdi watpyour`

- Adjusted
  - …`at euieno amtwfpvldoshs iretcra enogorsdi watpybl`



Security: Convergence to Input Distribution

- Markov chains will have considerably more convoluted convergence properties

Different implementations will introduce different biases.

Security

- This is not suitable as a replacement for a proper encryption!
- It may be suitable for steganographic applications though.