BILL
RECORD THIS LECTURE

June 9, 2021
Gen Sub Cipher: How to Really Crack

June 9, 2021
General Substitution Cipher

Def Gen Sub Cipher with perm $f$ on $\{0, \ldots, 25\}$.  
1. Encrypt via $x \rightarrow f(x)$.  
2. Decrypt via $x \rightarrow f^{-1}(x)$. 
Terminology: 1-Gram, 2-Gram, 3-Gram

**Notation** Let $T$ be a text.
Terminology: 1-Gram, 2-Gram, 3-Gram

**Notation** Let $T$ be a text.

1. The **1-grams** of $T$ are just the letters in $T$, counting repeats.
Terminology: 1-Gram, 2-Gram, 3-Gram

Notation Let $T$ be a text.

1. The 1-grams of $T$ are just the letters in $T$, counting repeats.
2. The 2-grams of $T$ are just the contiguous pairs of letters in $T$, counting repeats. Also called bigrams.
Terminology: 1-Gram, 2-Gram, 3-Gram

**Notation** Let $T$ be a text.

1. The **1-grams** of $T$ are just the letters in $T$, counting repeats.
2. The **2-grams** of $T$ are just the contiguous pairs of letters in $T$, counting repeats. Also called **bigrams**.
3. The **3-grams** of $T$ you can guess. Also called **trigrams**.
Terminology: 1-Gram, 2-Gram, 3-Gram

**Notation** Let $T$ be a text.

1. The **1-grams** of $T$ are just the letters in $T$, counting repeats.
2. The **2-grams** of $T$ are just the contiguous pairs of letters in $T$, counting repeats. Also called **bigrams**.
3. The **3-grams** of $T$ you can guess. Also called **trigrams**.
4. One usually talks about the freq of $n$-grams.
Example of 1-Grams

Let the text be:

*Ever notice how sometimes people use math words incorrectly?*
Example of 1-Grams

Let the text be:

Ever notice how sometimes people use math words incorrectly?

The following 1-grams occur 1 time: a,d,u,v,y.
Example of 1-Grams

Let the text be:

**Ever notice how sometimes people use math words incorrectly?**

The following 1-grams occur 1 time: a,d,u,v,y.
The following 1-grams occur 2 times: h,l,n,p,w.
Example of 1-Grams

Let the text be:

*Ever notice how sometimes people use math words incorrectly?*

The following 1-grams occur 1 time: a,d,u,v,y.
The following 1-grams occur 2 times: h,l,n,p,w.
The following 1-grams occur 3 times: c,i,m.
Example of 1-Grams

Let the text be:

**Ever notice how sometimes people use math words incorrectly?**

The following 1-grams occur 1 time: a,d,u,v,y.
The following 1-grams occur 2 times: h,l,n,p,w.
The following 1-grams occur 3 times: c,i,m.
The following 1-grams occur 4 times: r,s,t.
Example of 1-Grams

Let the text be:

*Ever notice how sometimes people use math words incorrectly?*

The following 1-grams occur 1 time: a,d,u,v,y.
The following 1-grams occur 2 times: h,l,n,p,w.
The following 1-grams occur 3 times: c,i,m.
The following 1-grams occur 4 times: r,s,t.
The following 1-gram occurs 6 times: o.
Example of 1-Grams

Let the text be:

*Ever notice how sometimes people use math words incorrectly?*

The following 1-grams occur 1 time: a,d,u,v,y.
The following 1-grams occur 2 times: h,l,n,p,w.
The following 1-grams occur 3 times: c,i,m.
The following 1-grams occur 4 times: r,s,t.
The following 1-gram occurs 6 times: o.
The following 1-gram occurs 9 times: e.
Example of 2-Grams

Let the text be:

**Ever notice how sometimes people use math words incorrectly?**
Example of 2-Grams

Let the text be:

**Ever notice how sometimes people use math words incorrectly?**

The following 2-grams occur 2 times: me, or.
Example of 2-Grams

Let the text be:

*Ever notice how sometimes people use math words incorrectly?*

The following 2-grams occur 2 times: me, or.

The following 2-grams occur 1 time: ev, ve, er, rn, no, ot, ti, ic, eh, ho, ow, ws, so, et, ti, im, es, sp, pe, eo, op, pl, le, eu, us, se, em, ma, at, th, hw, wo, ds, in, nc, co, rr, re, ec, ct, tl, ly.
**Notation** Let $\sigma$ be a perm and $T$ a text.
Notation and Parameter for a Family of Algorithms

**Notation** Let $\sigma$ be a perm and $T$ a text.

1. $f_E$ is freq of $n$-grams. It is a $26^n$ long vector. (Formally we should use $f_E(n)$. We omit the $n$. The value of $n$ will be clear from context.)

2. $\sigma(T)$ is taking $T$ and applying $\sigma$ to it. If $\sigma^{-1}$ was used to encrypt, then $\sigma(T)$ will be English!

3. $f_{\sigma(T)}$ is the $26^n$-long vector of freq's of $n$-grams in $\sigma(T)$.

4. $I$ and $R$ will be parameters we discuss later. $I$ stands for Iterations and will be large (like 2000). $R$ stands for Redos and will be small (like 5).
Notation and Parameter for a Family of Algorithms

**Notation** Let $\sigma$ be a perm and $T$ a text.

1. $f_E$ is freq of $n$-grams. It is a $26^n$ long vector. (Formally we should use $f_E(n)$. We omit the $n$. The value of $n$ will be clear from context.)

2. $\sigma(T)$ is taking $T$ and applying $\sigma$ to it. If $\sigma^{-1}$ was used to encrypt, then $\sigma(T)$ will be English!
Notation and Parameter for a Family of Algorithms

**Notation** Let $\sigma$ be a perm and $T$ a text.

1. $f_E$ is freq of $n$-grams. It is a $26^n$ long vector. (Formally we should use $f_E(n)$. We omit the $n$. The value of $n$ will be clear from context.)

2. $\sigma(T)$ is taking $T$ and applying $\sigma$ to it. If $\sigma^{-1}$ was used to encrypt, then $\sigma(T)$ will be English!

3. $f_{\sigma(T)}$ is the $26^n$-long vector of freq’s of $n$-grams in $\sigma(T)$. 

$I$ and $R$ will be parameters we discuss later. $I$ stands for Iterations and will be large (like 2000). $R$ stands for Redos and will be small (like 5).
Notation and Parameter for a Family of Algorithms

**Notation** Let $\sigma$ be a perm and $T$ a text.

1. $f_E$ is freq of $n$-grams. It is a $26^n$ long vector. (Formally we should use $f_E(n)$. We omit the $n$. The value of $n$ will be clear from context.)
2. $\sigma(T)$ is taking $T$ and applying $\sigma$ to it. If $\sigma^{-1}$ was used to encrypt, then $\sigma(T)$ will be English!
3. $f_{\sigma(T)}$ is the $26^n$-long vector of freq’s of $n$-grams in $\sigma(T)$.
4. I and R will be parameters we discuss later.
Notation and Parameter for a Family of Algorithms

Notation Let $\sigma$ be a perm and $T$ a text.

1. $f_E$ is freq of $n$-grams. It is a $26^n$ long vector. (Formally we should use $f_E(n)$. We omit the $n$. The value of $n$ will be clear from context.)

2. $\sigma(T)$ is taking $T$ and applying $\sigma$ to it. If $\sigma^{-1}$ was used to encrypt, then $\sigma(T)$ will be English!

3. $f_{\sigma(T)}$ is the $26^n$-long vector of freq’s of $n$-grams in $\sigma(T)$.

4. I and R will be parameters we discuss later. I stands for Iterations and will be large (like 2000).
Notation and Parameter for a Family of Algorithms

Notation Let $\sigma$ be a perm and $T$ a text.

1. $f_E$ is freq of $n$-grams. It is a $26^n$ long vector. (Formally we should use $f_E(n)$. We omit the $n$. The value of $n$ will be clear from context.)

2. $\sigma(T)$ is taking $T$ and applying $\sigma$ to it. If $\sigma^{-1}$ was used to encrypt, then $\sigma(T)$ will be English!

3. $f_{\sigma(T)}$ is the $26^n$-long vector of freq’s of $n$-grams in $\sigma(T)$.

4. I and R will be parameters we discuss later. I stands for Iterations and will be large (like 2000). R stands for Redos and will be small (like 5).
Stats for 1-Gram, 2-Gram, 3-Gram, 4-Gram

1. 1-grams:
\[ f_E \cdot f_E \sim 0.065. \]

2. 2-grams:
\[ f_E \cdot f_E \sim 0.0067. \]

3. 3-grams:
\[ f_E \cdot f_E \sim 0.0011. \]

4. 4-grams:
\[ f_E \cdot f_E \sim 0.00023. \]
Stats for 1-Gram, 2-Gram, 3-Gram, 4-Gram

1. 1-grams: $f_E \cdot f_E \sim 0.065$. 
Stats for 1-Gram, 2-Gram, 3-Gram, 4-Gram

1. 1-grams: \( f_E \cdot f_E \sim 0.065 \).
2. 2-grams: \( f_E \cdot f_E \sim 0.0067 \).
Stats for 1-Gram, 2-Gram, 3-Gram, 4-Gram

1. 1-grams: $f_E \cdot f_E \sim 0.065$.
2. 2-grams: $f_E \cdot f_E \sim 0.0067$.
3. 3-grams: $f_E \cdot f_E \sim 0.0011$. 
Stats for 1-Gram, 2-Gram, 3-Gram, 4-Gram

1. 1-grams: \( f_E \cdot f_E \sim 0.065 \).
2. 2-grams: \( f_E \cdot f_E \sim 0.0067 \).
3. 3-grams: \( f_E \cdot f_E \sim 0.0011 \).
4. 4-grams: \( f_E \cdot f_E \sim 0.00023 \).
Contrast Shift to Gen Sub

To crack shift went through all 26 shifts $\sigma$:

1. If $f(\sigma(T)) \cdot f(E)$ is large then $\sigma$ is correct shift. Large $\sim 0.065$.

2. If $f(\sigma(T)) \cdot f(E)$ is small then $\sigma$ is incorrect shift. Small.

3. Important. Will always be large or small. So we have a gap.

Lets try this with gen sub, ignoring the issue of 26! perms.

To crack gen sub shift went through all 26! perm $\sigma$:

1. If $f(\sigma(T)) \cdot f(E)$ is large then $\sigma$ is correct perm. Large $\sim 0.065$.

2. If $f(\sigma(T)) \cdot f(E)$ is small then $\sigma$ is incorrect perm. Small.

Hmmm?

3. We have a problem. If $\sigma$ only changed a few letters around, then likely $f(E) \cdot f(\sigma(T))$ will be large. We do not have a gap!

What to do?
To crack shift went through all 26 shifts $\sigma$:

1. If $f_{\sigma(T)} \cdot f_E$ is large then $\sigma$ is correct shift. Large $\sim 0.065$.

2. If $f_{\sigma(T)} \cdot f_E$ is small then $\sigma$ is incorrect shift. Small.

3. Important: Will always be large or small. So we have a gap.

Let's try this with gen sub, ignoring the issue of $26!$ perms.

To crack gen sub shift went through all $26!$ permutations $\sigma$:

1. If $f_{\sigma(T)} \cdot f_E$ is large then $\sigma$ is correct perm. Large $\sim 0.065$.

2. If $f_{\sigma(T)} \cdot f_E$ is small then $\sigma$ is incorrect perm. Small.

Hmmm?

3. We have a problem. If $\sigma$ only changed a few letters around, then likely $f_E \cdot f_{\sigma(T)}$ will be large. We do not have a gap!

What to do?
Contrast Shift to Gen Sub

To crack shift went through all 26 shifts $\sigma$:

1. If $f_{\sigma(T)} \cdot f_E$ is large then $\sigma$ is correct shift. Large $\sim 0.065$.
2. If $f_{\sigma(T)} \cdot f_E$ is small then $\sigma$ is incorrect shift. Small $\sim 0.035$.

Hmmm?

3. We have a problem. If $\sigma$ only changed a few letters around, then likely $f_E \cdot f_{\sigma(T)}$ will be large. We do not have a gap!

What to do?
Contrast Shift to Gen Sub

To crack shift went through all 26 shifts $\sigma$:

1. If $f_{\sigma(T)} \cdot f_E$ is large then $\sigma$ is correct shift. Large $\sim 0.065$.
2. If $f_{\sigma(T)} \cdot f_E$ is small then $\sigma$ is incorrect shift. Small $\sim 0.035$.
3. **Important** Will always be large or small. So we have a gap.
Contrast Shift to Gen Sub

To crack shift went through all 26 shifts $\sigma$:

1. If $f_{\sigma(T)} \cdot f_E$ is large then $\sigma$ is correct shift. Large $\sim 0.065$.
2. If $f_{\sigma(T)} \cdot f_E$ is small then $\sigma$ is incorrect shift. Small $\sim 0.035$.
3. **Important** Will always be large or small. So we have a gap.

Lets try this with gen sub, ignoring the issue of 26! perms.
To crack gen sub shift went through all 26! perm $\sigma$:
Contrast Shift to Gen Sub

To crack shift went through all 26 shifts \( \sigma \):

1. If \( f_{\sigma(T)} \cdot f_E \) is large then \( \sigma \) is correct shift. Large \( \sim 0.065 \).
2. If \( f_{\sigma(T)} \cdot f_E \) is small then \( \sigma \) is incorrect shift. Small \( \sim 0.035 \).
3. Important Will always be large or small. So we have a gap.

Lets try this with gen sub, ignoring the issue of 26! perms.

To crack gen sub shift went through all 26! perm \( \sigma \):

1. If \( f_{\sigma(T)} \cdot f_E \) is large then \( \sigma \) is correct perm. Large \( \sim 0.065 \).
Contrast Shift to Gen Sub

To crack shift went through all 26 shifts $\sigma$:

1. If $f_{\sigma(T)} \cdot f_E$ is large then $\sigma$ is correct shift. Large $\sim 0.065$.
2. If $f_{\sigma(T)} \cdot f_E$ is small then $\sigma$ is incorrect shift. Small $\sim 0.035$.
3. Important Will always be large or small. So we have a gap.

Lets try this with gen sub, ignoring the issue of 26! perms.
To crack gen sub shift went through all 26! perm $\sigma$:

1. If $f_{\sigma(T)} \cdot f_E$ is large then $\sigma$ is correct perm. Large $\sim 0.065$.
2. If $f_{\sigma(T)} \cdot f_E$ is small then $\sigma$ is incorrect perm. Small.
Contrast Shift to Gen Sub

To crack shift went through all 26 shifts $\sigma$:

1. If $f_{\sigma(T)} \cdot f_E$ is large then $\sigma$ is correct shift. Large $\sim 0.065$.
2. If $f_{\sigma(T)} \cdot f_E$ is small then $\sigma$ is incorrect shift. Small $\sim 0.035$.
3. Important Will always be large or small. So we have a gap.

Lets try this with gen sub, ignoring the issue of 26! perms.
To crack gen sub shift went through all 26! perm $\sigma$:

1. If $f_{\sigma(T)} \cdot f_E$ is large then $\sigma$ is correct perm. Large $\sim 0.065$.
2. If $f_{\sigma(T)} \cdot f_E$ is small then $\sigma$ is incorrect perm. Small. Hmm?
Contrast Shift to Gen Sub

To crack shift went through all 26 shifts $\sigma$:

1. If $f_{\sigma(T)} \cdot f_E$ is large then $\sigma$ is correct shift. Large $\sim 0.065$.
2. If $f_{\sigma(T)} \cdot f_E$ is small then $\sigma$ is incorrect shift. Small $\sim 0.035$.
3. **Important** Will always be large or small. So we have a gap.

Lets try this with gen sub, ignoring the issue of $26!$ perms.
To crack gen sub shift went through all $26!$ perm $\sigma$:

1. If $f_{\sigma(T)} \cdot f_E$ is large then $\sigma$ is correct perm. Large $\sim 0.065$.
2. If $f_{\sigma(T)} \cdot f_E$ is small then $\sigma$ is incorrect perm. Small. Hmmm?
3. We have a problem. If $\sigma$ only changed a few letters around, then likely $f_E \cdot f_{\sigma(T)}$ will be large. We **do not** have a gap!

What to do?
What to do if there is no Gap?

1. Use n-grams instead of 1-grams. This does not close the Gap but will help anyway.
2. Rather than view the Is-English program as a YES-NO, view it as comparative: $T_1$ looks more like English than $T_2$. 
What to do if there is no Gap?

1. Use $n$-grams instead of 1-grams. This does not close the Gap but will help anyway.
What to do if there is no Gap?

1. Use $n$-grams instead of 1-grams. This does not close the Gap but will help anyway.

2. Rather than view the Is-English program as a YES-NO, view it as comparative:
   
   $T_1$ looks more like English than $T_2$. 
Input $T$. Find Freq of 1-grams and $n$-grams.
Input $T$. Find Freq of 1-grams and $n$-grams.

$\sigma_{\text{init}}$ is perm that maps most freq to $e$, etc. Uses 1-gram freq.
Input $T$. Find Freq of 1-grams and $n$-grams.

$\sigma_{\text{init}}$ is perm that maps most freq to e, etc. Uses 1-gram freq.

For $r = 1$ to $R$ ($R$ is small, about 5)
Input $T$. Find Freq of 1-grams and $n$-grams.

$\sigma_{init}$ is perm that maps most freq to $e$, etc. Uses 1-gram freq.

For $r = 1$ to $R$ ($R$ is small, about 5)

$$\sigma_r \leftarrow \sigma_{init}$$
Input $T$. Find Freq of 1-grams and $n$-grams.

$\sigma_{\text{init}}$ is perm that maps most freq to $e$, etc. Uses 1-gram freq.

For $r = 1$ to $R$ ($R$ is small, about 5)

$$\sigma_r \leftarrow \sigma_{\text{init}}$$

For $i = 1$ to $I$ ($I$ is large, about 2000)
\textit{n-Gram Algorithm}

Input $T$. Find Freq of 1-grams and $n$-grams.

$\sigma_{\text{init}}$ is perm that maps most freq to $e$, etc. Uses 1-gram freq.

For $r = 1$ to $R$ ($R$ is small, about 5)

\begin{align*}
\sigma_r &\leftarrow \sigma_{\text{init}} \\
&\text{For } i = 1 \text{ to } I \text{ (}I\text{ is large, about 2000)} \\
&\quad \text{Pick } j, k \in \{0, \ldots, 25\} \text{ at Random.}
\end{align*}
**n-Gram Algorithm**

Input $T$. Find Freq of 1-grams and $n$-grams. 

$\sigma_{\text{init}}$ is perm that maps most freq to e, etc. Uses 1-gram freq.

For $r = 1$ to $R$ ($R$ is small, about 5)

$\quad \sigma_r \leftarrow \sigma_{\text{init}}$

For $i = 1$ to $I$ ($I$ is large, about 2000)

$\quad$ Pick $j, k \in \{0, \ldots, 25\}$ at Random.
$\quad$ Let $\sigma'$ be $\sigma_r$ with $j, k$ swapped
\textit{n-Gram Algorithm}

Input $T$. Find Freq of 1-grams and $n$-grams. 

$\sigma_{\text{init}}$ is perm that maps most freq to \textit{e}, etc. Uses 1-gram freq.

For $r = 1$ to $R$ ($R$ is small, about 5)

\begin{align*}
\sigma_r & \leftarrow \sigma_{\text{init}} \\
\text{For } i = 1 \text{ to } I \text{ (I is large, about 2000)} & \\
\text{Pick } j, k \in \{0, \ldots, 25\} \text{ at Random.} & \\
\text{Let } \sigma' \text{ be } \sigma_r \text{ with } j, k \text{ swapped} & \\
\text{If } f_{\sigma'(T)} \cdot f_E > f_{\sigma_r(T)} \cdot f_E \text{ then } \sigma_r & \leftarrow \sigma'
\end{align*}
$n$-Gram Algorithm

Input $T$. Find Freq of 1-grams and $n$-grams. 
$\sigma_{\text{init}}$ is perm that maps most freq to $e$, etc. Uses 1-gram freq.

For $r = 1$ to $R$ ($R$ is small, about 5)

$\sigma_r \leftarrow \sigma_{\text{init}}$

For $i = 1$ to $I$ ($I$ is large, about 2000)

Pick $j, k \in \{0, \ldots, 25\}$ at Random.

Let $\sigma'$ be $\sigma_r$ with $j, k$ swapped

If $f_{\sigma'(T)} \cdot f_E > f_{\sigma_r(T)} \cdot f_E$ then $\sigma_r \leftarrow \sigma'$

Candidates for $\sigma$ are $\sigma_1, \ldots, \sigma_R$
Input \( T \). Find Freq of 1-grams and \( n \)-grams. 
\( \sigma_{\text{init}} \) is perm that maps most freq to \( e \), etc. Uses 1-gram freq.

For \( r = 1 \) to \( R \) (\( R \) is small, about 5)

\[
\sigma_r \leftarrow \sigma_{\text{init}}
\]

For \( i = 1 \) to \( I \) (\( I \) is large, about 2000)

Pick \( j, k \in \{0, \ldots, 25\} \) at Random.

Let \( \sigma' \) be \( \sigma_r \) with \( j, k \) swapped

If \( f_{\sigma'(T)} \cdot f_E > f_{\sigma_r(T)} \cdot f_E \) then \( \sigma_r \leftarrow \sigma' \)

Candidates for \( \sigma \) are \( \sigma_1, \ldots, \sigma_R \)

Pick the \( \sigma_r \) with min \( \text{good}_r \) or have human look at all \( \sigma_r(T) \)
An old question:
What came first, the chicken or the egg?
An old question:
What came first, the chicken or the egg?

**Our Problem** We need parameters I and R so the answer looks like English. But we then need a notion of Is English that does not use a gap. Need a program to tell us that it looks like English.
Finding Parameters: A Chicken-and-Egg Problem

An old question:
What came first, the chicken or the egg?

Our Problem We need parameters I and R so the answer looks like English. But we then need a notion of Is English that does not use a gap. Need a program to tell us that it looks like English.

We Trebekked It!
Finding Parameters: A Chicken-and-Egg Problem

An old question:
What came first, the chicken or the egg?

Our Problem We need parameters I and R so the answer looks like English. But we then need a notion of Is English that does not use a gap. Need a program to tell us that it looks like English.

We Trebekked It!
On the TV show JEOPARDY Alex Trebek (before he died) gives you the answer and you have to figure out the question.
An old question:

**What came first, the chicken or the egg?**

**Our Problem** We need parameters $I$ and $R$ so the answer looks like English. But we then need a notion of *Is English* that does not use a gap. Need a program to tell us that it looks like English.

**We Trebekked It!**

On the TV show JEOPARDY Alex Trebek (before he died) gives you the *answer* and you have to figure out the *question*.

Same here.
An old question:
What came first, the chicken or the egg?

**Our Problem** We need parameters $I$ and $R$ so the answer looks like English. But we then need a notion of Is English that does not use a gap. Need a program to tell us that it looks like English.

**We Trebekked It!**
On the TV show JEOPARDY Alex Trebek (before he died) gives you the *answer* and you have to figure out the *question*.

Same here.

We find the parameters for texts where we know the answers.
Finding the Parameters

Do the following a large number of times:

1. Take a text \( T \) of \( \sim 10,000 \) characters.
2. Take a random perm \( \sigma \).
3. Compute \( \sigma(T) \). (Note - We know \( \sigma \) and \( T \)).
4. Run the \( n \)-gram algorithm but with no bound on the number of iterations. Stop when either
   4.1 Get original text \( T \), or
   4.2 Swaps do not improve how close to English (could be in local min).

5. Keep track of how many iterations suffice and how many redos suffice.
Finding the Parameters

Do the following a large number of times:

1. Take a text $T$ of $\sim 10,000$ characters.
Finding the Parameters

Do the following a large number of times:

1. Take a text $T$ of $\sim 10,000$ characters.
2. Take a random perm $\sigma$.
Finding the Parameters

Do the following a large number of times:

1. Take a text $T$ of $\sim 10,000$ characters.
2. Take a random perm $\sigma$.
3. Compute $\sigma(T)$. (Note- We know $\sigma$ and $T$)
Finding the Parameters

Do the following a large number of times:

1. Take a text $T$ of $\sim 10,000$ characters.
2. Take a random perm $\sigma$.
3. Compute $\sigma(T)$. (Note- We know $\sigma$ and $T$)
4. Run the $n$-gram algorithm but with no bound on the number of iterations. Stop when either
Finding the Parameters

Do the following a large number of times:

1. Take a text $T$ of $\sim 10,000$ characters.
2. Take a random perm $\sigma$.
3. Compute $\sigma(T)$. (Note- We know $\sigma$ and $T$)
4. Run the $n$-gram algorithm but with no bound on the number of iterations. Stop when either
   4.1 Get original text $T$, or
Finding the Parameters

Do the following a large number of times:

1. Take a text $T$ of $\sim 10,000$ characters.
2. Take a random perm $\sigma$.
3. Compute $\sigma(T)$. (Note- We know $\sigma$ and $T$)
4. Run the $n$-gram algorithm but with no bound on the number of iterations. Stop when either
   4.1 Get original text $T$, or
   4.2 Swaps do not improve how close to English (could be in local min). In this case try again.
Finding the Parameters

Do the following a large number of times:

1. Take a text $T$ of $\sim 10,000$ characters.
2. Take a random perm $\sigma$.
3. Compute $\sigma(T)$. (Note- We know $\sigma$ and $T$)
4. Run the $n$-gram algorithm but with no bound on the number of iterations. Stop when either
   4.1 Get original text $T$, or
   4.2 Swaps do not improve how close to English (could be in local min). In this case try again.
5. Keep track of how many iterations suffice and how many redos suffice.
UMCP ugrad CS major David Zhen worked with me on this over the summer.
UMCP ugrad CS major David Zhen worked with me on this over the summer.

The next three slides show the parameters he found.
UMCP ugrad CS major David Zhen worked with me on this over the summer.

The next three slides show the parameters he found.

He used a Mac-Book Pro with 2.2 Ghz 6-core Intel Core i7 processor and 16 GB of RAM.
UMCP ugrad CS major David Zhen worked with me on this over the summer.

The next three slides show the parameters he found.

He used a Mac-Book Pro with 2.2 Ghz 6-core Intel Core i7 processor and 16 GB of RAM.

In English: a normal computer that an ugrad can buy and use.
David Zhen Found the Parameters

UMCP ugrad CS major David Zhen worked with me on this over the summer.

The next three slides show the parameters he found.

He used a Mac-Book Pro with 2.2 Ghz 6-core Intel Core i7 processor and 16 GB of RAM.

In English: a normal computer that an ugrad can buy and use.

He ran the program to find parameters on 150 texts of size approx 10,000 characters:
UMCP ugrad CS major David Zhen worked with me on this over the summer.

The next three slides show the parameters he found.

He used a Mac-Book Pro with 2.2 Ghz 6-core Intel Core i7 processor and 16 GB of RAM.

In English: a normal computer that an ugrad can buy and use.

He ran the program to find parameters on 150 texts of size approx 10,000 characters:

For each text he generated 1 random perm (will rerun with more later).
Parameters for $n$-Grams

1-grams: Nothing worked

2-grams: Nothing worked

3-grams: $I = 2000$, $R = 4$ worked. Took $\leq 2$ minutes to crack.

4-grams: $I = 2000$, $R = 8$, Took around 6 minutes to crack.
Parameters for $n$-Grams

1-grams Nothing worked
Parameters for $n$-Grams

1-grams Nothing worked
2-grams Nothing worked

$\mathbf{i} = 2000$, $\mathbf{R} = 4$ worked. Took $\leq 2$ minutes to crack.

4-grams $\mathbf{i} = 2000$, $\mathbf{R} = 8$, Took around 6 minutes to crack.
Parameters for $n$-Grams

1-grams Nothing worked
2-grams Nothing worked
3-grams $I = 2000$, $R = 4$ worked. Took $\leq 2$ minutes to crack.
Parameters for $n$-Grams

- **1-grams** Nothing worked
- **2-grams** Nothing worked
- **3-grams** $I = 2000$, $R = 4$ worked. Took $\leq 2$ minutes to crack.
- **4-grams** $I = 2000$, $R = 8$, Took around 6 minutes to crack.