Stanford University

ENGR 76

Information Science and Engineering

Lecture 2: Source Coding I

Introduction and Prefix-Free Codes

Siddharth Chandak

Logistics

Office Hours

- **Instructor Office Hours:**
  - Siddharth: Wednesday 11:30am - 1pm - Packard 107
    - **Lectures and mini-PSets**
  - Prof. Ozgur: By appointment (schedule via email)
- **TA Office Hours**:
  - Any questions regarding the mini-PSet or the project assignments
  - Tuesday 10:30am - 12:30pm - Packard 106
  - Thursday 2pm - 4pm - Packard 104

Discussion Session

- **Mini-PSet Discussion Session:**
  - Monday 11am - noon in Room 60-109
  - Andy will discuss problems similar to the mini-PSet for 30-40 minutes

Submission Details

- **Mini-PSet**
  - Gradescope online assignments
  - Untimed and will allow multiple submissions
  - Released on Thursday and due next Wednesday at 11:59pm
- **Project Assignments**
  - Python notebooks (setup in Project 0)
  - Will be uploaded to Canvas and need to be submitted to Gradescope
  - Released on Friday and due next Friday at 11:59pm

Course Announcements and Reminders

Reminders

- **Please fill the entry survey!**
- Make sure you have been added to Canvas, Ed, and Gradescope. If not, reach out to us ASAP.
- Mini-PSet 1 will be released today

Project 0

- Project 0 will be released tomorrow

ENGR 76 Project 0: Getting started

Due: Friday, January 16, 2026 at 11:59pm.

The task this week is to "get started." Our goals are to:

1. Install everything we need for Project 1.
2. Get familiar with the NumPy library.
3. Learn to load images.
4. Have a first look at the sizes of compressed and uncompressed images.

Start Early!!!

Lecture 2: Introduction to Source Coding

Let's start with some examples!

Example I

- You flip a coin 10 times. For each flip, you either get a Heads (**H**) or a Tails (**T**).
- You write down this sequence, e.g., HTTHHTHTTT
- Now you want to express this as a sequence of bits.
  - What will you map H to?
  - What will you map T to?

Example I

- You flip a coin 10 times. For each flip, you either get a Heads (**H**) or a Tails (**T**).
- You write down this sequence, e.g., HTTHHTHTTT
- Now you want to express this as a sequence of bits.
  - What will you map H to? — **0**
  - What will you map T to? — **1**

Example I

- Sequence - HTTHHTHTTT
- H is mapped to 0
- T is mapped to 1
- How will the sequence be expressed?

Example I

- Sequence - HTTHHTHTTT
- H is mapped to 0
- T is mapped to 1
- How will the sequence be expressed?
  - **0110010111**

Example I

- Now you give your friend give the bit sequence 0110010111?
- What else do they need to know to decode?

Example I

- Now you give your friend give the bit sequence 0110010111?
- What else do they need to know to decode?
  - In this scenario, they need to know that H is mapped to 0 and T is mapped to 1
- How will they go back to the sequence of H and T from the bit sequence 0110010111?

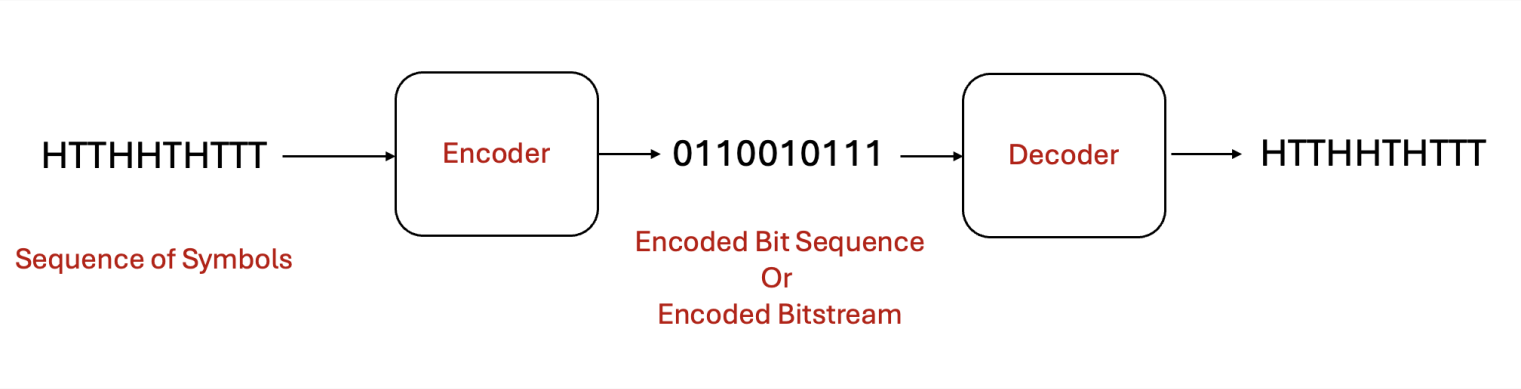
Example I

- **Alphabet** $\mathcal{X}$ - Set of symbols
- M: the number of symbols in alphabet
- **Codeword**: Each symbol is mapped to a codeword, i.e., for each $x \in \mathcal{X}$, codeword $c(x)$ is the binary sequence corresponding to symbol x
- **Length of Codeword**: For each symbol, length of its codeword $\ell(x)$ is the number of bits in its codeword
- **Code**: A code is a mapping from $\mathcal{X}$ to the set of binary sequences

Example I

- **Alphabet** $\mathcal{X}$ - Set of symbols
  - {H,T}
- M: the number of symbols in alphabet
  - $M = 2$
- **Codeword**: Each symbol is mapped to a codeword, i.e., for each $x \in \mathcal{X}$, codeword $c(x)$ is the binary sequence corresponding to symbol x
  - $c(\text{H}) = 0$ (codeword for H is 0)
- **Length of Codeword**: For each symbol, length of its codeword $\ell(x)$ is the number of bits in its codeword
  - $\ell(\text{H}) = 1, \ell(\text{T}) = 1$
- **Code**: A code is a mapping from $\mathcal{X}$ to the set of binary sequences
  - Dictionary of codewords (dictionary of mappings from symbols to codewords)

Example I


HTTHHTHTTT
Encoder
0110010111
Decoder
HTTHHTHTTT
Sequence of Symbols
Encoded Bit Sequence
Or
Encoded Bitstream

Example I

- Why did we use the code $(\mathsf{H} \to 0, \mathsf{T} \to 1)$? Why not $(\mathsf{H} \to 00, \mathsf{T} \to 11)$?

Example I

- Why did we use the code $(\mathsf{H} \to 0, \mathsf{T} \to 1)$? Why not $(\mathsf{H} \to 00, \mathsf{T} \to 11)$?
  - Want to use as few bits as possible.

Example II

- Now you have 4 symbols A, B, C, D and you want to encode the sequence BDCDBABDDC.
  - Alphabet $\mathcal{X} = ?$
  - Number of symbols $M = ?$
  - Code?
  - Length of codewords?
  - Encoded bit sequence?
  - How to decode?

Example II

- Now you have 4 symbols A, B, C, D and you want to encode the sequence BDCDBABDDC.
  - Alphabet $\mathcal{X} = \{\text{A,B,C,D}\}$
  - Number of symbols $M = 4$
  - Code: $(\text{A} \to 00, \text{B} \to 01, \text{C} \to 10, \text{D} \to 11)$
    - **Just one possible code!**
  - Length of codewords: Each codeword is of length 2.
  - Encoded bit sequence: 01111011010001111110

Example II

- Code: $(\mathsf{A} \to 00, \mathsf{B} \to 01, \mathsf{C} \to 10, \mathsf{D} \to 11)$
- Want to minimize number of bits used...
- Why not use the code $(\mathsf{A} \to 0, \mathsf{B} \to 1, \mathsf{C} \to 0, \mathsf{D} \to 1)$?

Non-Singular Codes

Definition (Non-Singular Codes)

If $x \neq x'$, then $c(x) \neq c(x')$.

- Each symbol has a distinct codeword
  - $(\text{A} \to 00, \text{B} \to 01, \text{C} \to 10, \text{D} \to 11)$ is non-singular
  - $(\text{A} \to 0, \text{B} \to 1, \text{C} \to 0, \text{D} \to 1)$ is NOT non-singular
- We want our code to be non-singular

Fixed Length Codes

Fixed Length Codes

- A code where each codeword has the same length k

Fixed Length Codes

- A code where each codeword has the same length k
- For M symbols, what is the minimum possible k such that each symbol has a distinct codeword?

Fixed Length Codes

- A code where each codeword has the same length k
- For M symbols, what is the minimum possible k such that each symbol has a distinct codeword?
  - For $M = 2$, we saw $k = 1$
  - For $M = 4$, we saw $k = 2$
  - For general M?

Fixed Length Codes

- A code where each codeword has the same length k
- For M symbols, what is the minimum possible k such that each symbol has a distinct codeword?
  - $k = \lceil \log_2(M) \rceil$
  - $\lceil y \rceil$ is the smallest integer larger than or equal to y
    - for example, $\lceil 2.5 \rceil = 3$ and $\lceil 3 \rceil = 3$

Fixed Length Codes

- Each codeword has the same length $k = \lceil \log_2(M) \rceil$
  - For 3 symbols, we need 2 bits
  - $(\mathsf{A} \rightarrow 00, \mathsf{B} \rightarrow 01, \mathsf{C} \rightarrow 10)$

Decoding

- Decoding bit sequences encoded using fixed length codes is simple
- Algorithm:
  - Invert the code
  - Divide the bit sequence into blocks of k
  - Map each codeword to its corresponding symbol
- Example:
  - Code: $(\mathsf{A} \rightarrow 00, \mathsf{B} \rightarrow 01, \mathsf{C} \rightarrow 10)$
  - Bit sequence: 100001001000001

Decoding

- Example:
  - Code: $(\mathsf{A} \to 00, \mathsf{B} \to 01, \mathsf{C} \to 10)$
  - Bit sequence: 10000100100001
- Working of the algorithm:
  - Invert the code
    - $(00 \to \mathsf{A}, 01 \to \mathsf{B}, 10 \to \mathsf{C})$
  - Divide the bit sequence into blocks of k
    - 10 00 01 00 10 00 01
  - Map each codeword to its corresponding symbol
    - CABACAB

Are fixed length codes optimal?

- For 3 symbols, consider the fixed length code:
  - Code 1: $(\mathsf{A} \rightarrow 00, \mathsf{B} \rightarrow 01, \mathsf{C} \rightarrow 10)$
- Consider a different code:
  - Code 2: $(\mathsf{A} \rightarrow 00, \mathsf{B} \rightarrow 01, \mathsf{C} \rightarrow 1)$
- Which one will use fewer number of bits?

Are fixed length codes optimal?

- For 3 symbols, consider the fixed length code:
  - Code 1: $(\mathsf{A} \rightarrow 00, \mathsf{B} \rightarrow 01, \mathsf{C} \rightarrow 10)$
- Consider a different code:
  - Code 2: $(\mathsf{A} \rightarrow 00, \mathsf{B} \rightarrow 01, \mathsf{C} \rightarrow 1)$
- Which one will use fewer number of bits? — **Code 2**
- **Need variable length codes**

Question

Code 2: $(\mathsf{A} \to 00, \mathsf{B} \to 01, \mathsf{C} \to 1)$

- Why not $(\mathsf{A} \to 00, \mathsf{B} \to 01, \mathsf{C} \to 0)$?

Unique Decodability

Unique Decodability

Code: $(\mathsf{A} \to 00, \mathsf{B} \to 01, \mathsf{C} \to 0)$

- Non-singular ✓
- A "good" code ×

Unique Decodability

Code: $(\mathsf{A} \to 00, \mathsf{B} \to 01, \mathsf{C} \to 0)$

- Non-singular ✓
- A "good" code ×
  - Encoded bitstream: 00
  - Sequence of symbols - A or CC?

Unique Decodability

Definition (Unique Decodability)

A code is uniquely decodable if the encoding of every distinct sequence of symbols $X_1, X_2, \ldots$ (of arbitrary length) is distinct.

- Examples of codes which are NOT uniquely decodable:
  - $(\mathsf{A} \to 00, \mathsf{B} \to 01, \mathsf{C} \to 0)$
  - $(\mathsf{A} \to 0, \mathsf{B} \to 01, \mathsf{C} \to 1)$
  - $(\mathsf{A} \to 0, \mathsf{B} \to 01, \mathsf{C} \to 011, \mathsf{D} \to 10)$

Example

$$(\mathsf{A} \to 0, \mathsf{B} \to 01)$$

- Uniquely decodable? ✓
- Do we want to work with this? ×

Prefix-Free Codes

Prefix-Free Codes

Definition (Prefix-Free Codes)

No codeword is a prefix of another codeword.

- Examples:
  - $(\mathsf{A} \to 0, \mathsf{B} \to 0)$
  - $(\mathsf{A} \to 0, \mathsf{B} \to 01)$
  - $(\mathsf{A} \to 0, \mathsf{B} \to 10, \mathsf{C} \to 110, \mathsf{D} \to 111)$
  - $(\mathsf{A} \to 0, \mathsf{B} \to 1, \mathsf{C} \to 01)$

Prefix-Free Codes

Definition (Prefix-Free Codes)

No codeword is a prefix of another codeword.

- Examples:
  - $(\mathsf{A} \to 0, \mathsf{B} \to 0)$ ×
  - $(\mathsf{A} \to 0, \mathsf{B} \to 01)$ ×
  - $(\mathsf{A} \to 0, \mathsf{B} \to 10, \mathsf{C} \to 110, \mathsf{D} \to 111)$ ✓
  - $(\mathsf{A} \to 0, \mathsf{B} \to 1, \mathsf{C} \to 11)$ ×

Why Prefix-Free Codes?

- Uniquely decodable
- Instantaneously decodable

Decoding

Decoding algorithm:

- Invert code
- Start with buffer $\mathrm{buf}=$“ ”
- Loop:
  - Add next bit to buf
  - If buf is in inverted code:
    - Decode symbol
    - $\mathrm{buf}=$“ ”
  - Else:
    - Continue.

Example:

- $(\mathsf{A} \to 0, \mathsf{B} \to 10, \mathsf{C} \to 110, \mathsf{D} \to 111)$
- Bit sequence: 11000111

Prefix-Free Codes

Fact

For any uniquely decodable code, there exists a prefix-free code with the same codeword lengths.

- No loss of optimality in restricting to prefix-free codes

How to choose?

$$(\mathsf{A} \to 00, \mathsf{B} \to 01, \mathsf{C} \to 1)$$

vs

$$(\mathsf{A} \to 1, \mathsf{B} \to 01, \mathsf{C} \to 00)$$

Empirical Counts

How to choose?

Sequence: ABAAAABBAC

$$(\mathsf{A} \to 00, \mathsf{B} \to 01, \mathsf{C} \to 1)$$

vs

$$(\mathsf{A} \to 1, \mathsf{B} \to 01, \mathsf{C} \to 00)$$

How to choose?

Sequence: ABAAAABBAC

$$(\mathsf{A} \to 00, \mathsf{B} \to 01, \mathsf{C} \to 1) \quad \times$$

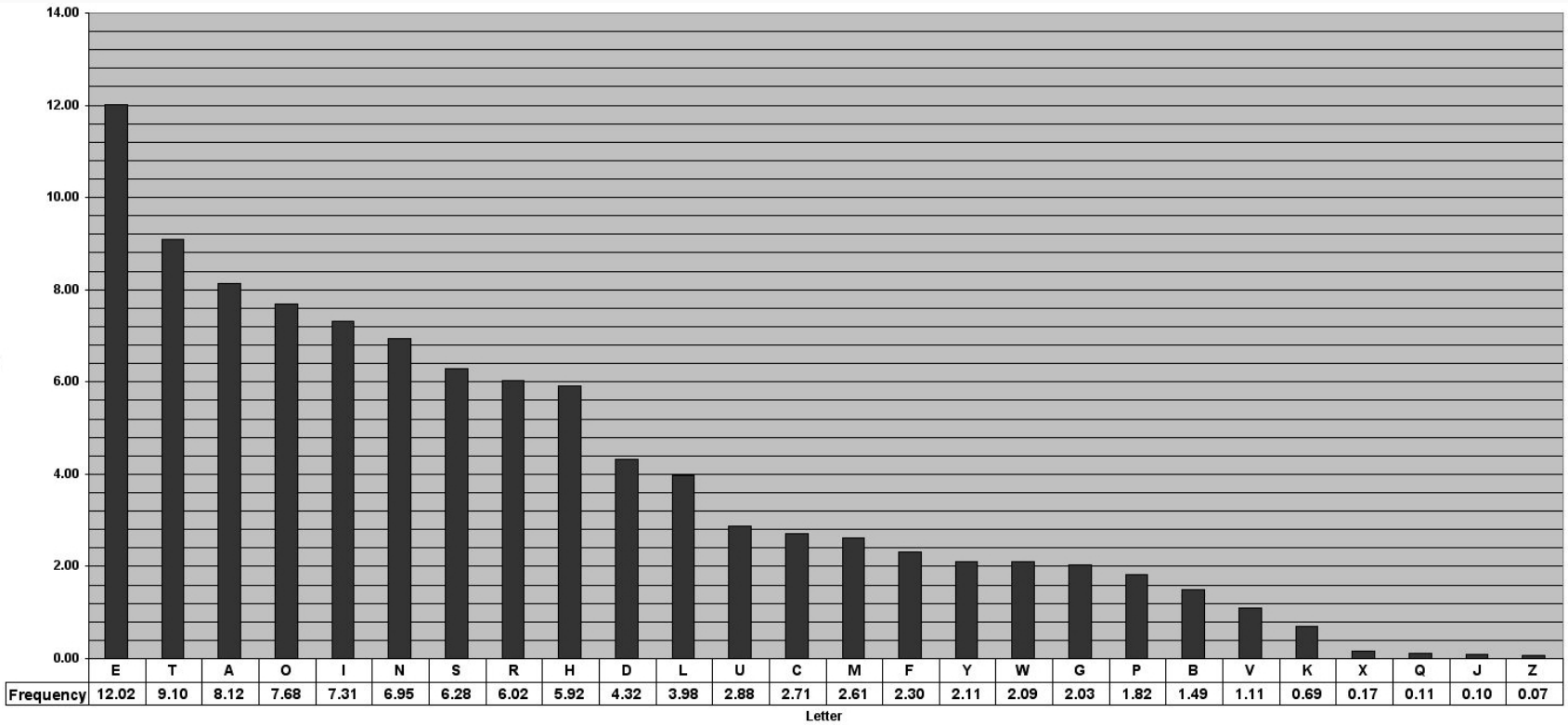
vs

$$(\mathsf{A} \to 1, \mathsf{B} \to 01, \mathsf{C} \to 00) \quad \checkmark$$

How to choose?

- Intuition: A symbol that occurs more times in the sequence should have a smaller codeword.

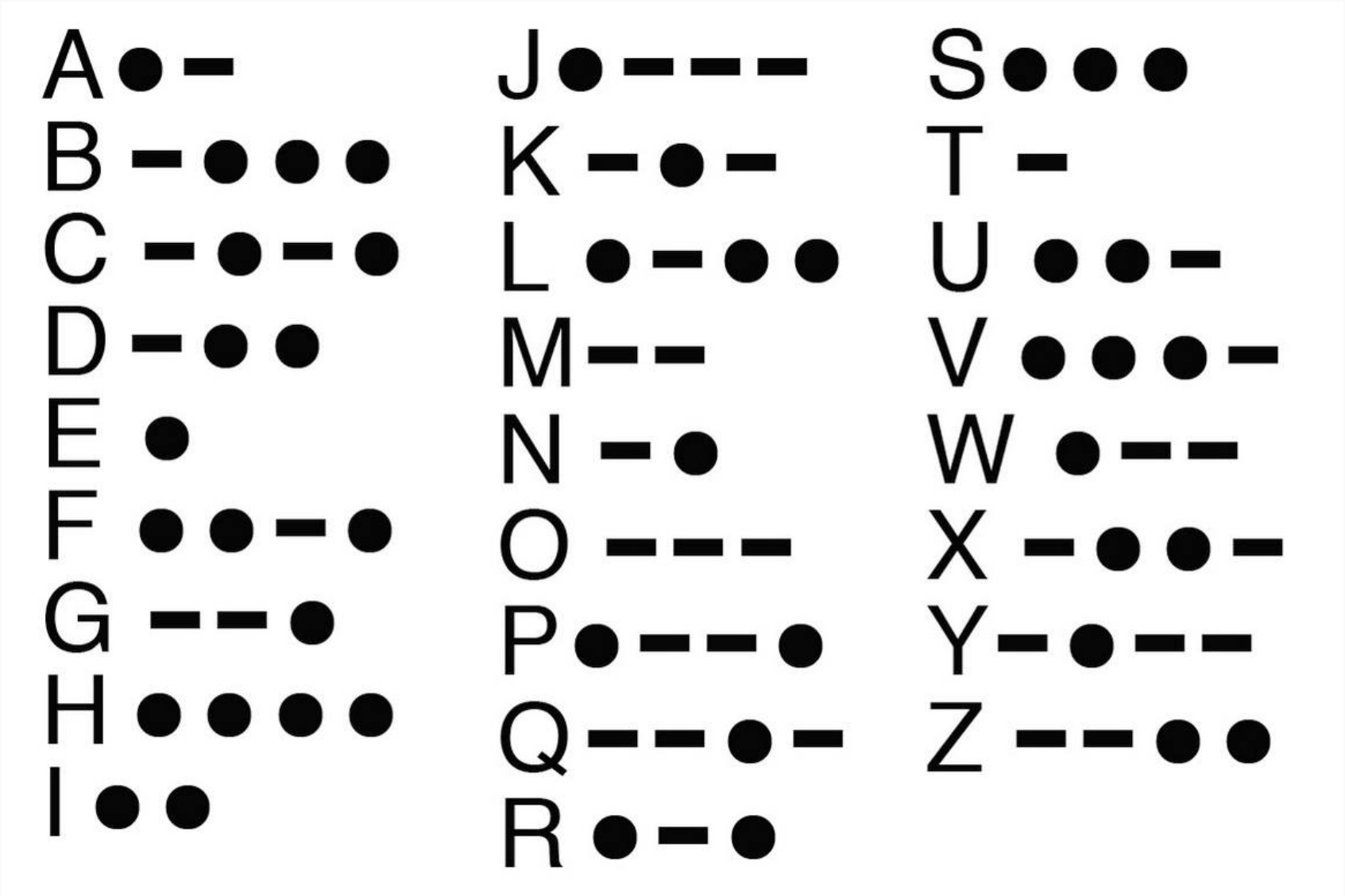
Example: English Alphabet



	E	T	A	O	I	N	S	R	H	D	L	U	C	M	F	Y	W	G	P	B	V	K	X	Q	J	Z
Frequency	12.02	9.10	8.12	7.68	7.31	6.95	6.28	6.02	5.92	4.32	3.98	2.88	2.71	2.61	2.30	2.11	2.09	2.03	1.82	1.49	1.11	0.69	0.17	0.11	0.10	0.07

Letter

Example: English Alphabet and Morse Code



How to choose?

- Intuition: A symbol that occurs more times in the sequence should have a smaller codeword.
- Formally?

Empirical Counts

- Sequence: $X_1 X_2 \ldots X_N$
- Each X_i is a symbol from the set $\mathcal{X}$
- $\boldsymbol{N}$: number of symbols in the sequence (length of sequence)
- **count** $\boldsymbol{n(x)}$: number of times symbol x appears in the sequence
- **frequency** $\boldsymbol{f(x)}$: frequency of symbol x in the sequence

$$f(x) = n(x)/N$$

Example: ABAAAABBAC

Empirical Counts

Example: ABAAAABBAC

- $N = 10$
- $n(\mathsf{A}) = 6,\ n(\mathsf{B}) = 3,\ n(\mathsf{C}) = 1$
- $f(\mathsf{A}) = 0.6,\ f(\mathsf{B}) = 0.3,\ f(\mathsf{C}) = 0.1$

Length of Encoded Bit Sequence

- Recall that $\ell(x)$ is length of codeword for symbol x
- Length of encoded bitstream?

Length of Encoded Bit Sequence

- Recall that $\ell(x)$ is length of codeword for symbol x
- Length of encoded bitstream?

$$\ell(X_1) + \ldots + \ell(X_N) = \sum_{x \in \mathcal{X}} \ell(x) n(x)$$

- Average number of bits per symbol for this sequence:

$$\frac{\text{Length of encoded bitstream}}{N} = \frac{\sum_{x \in \mathcal{X}} \ell(x) n(x)}{N} = \sum_{x \in \mathcal{X}} \ell(x) f(x)$$

- $\left(\sum_{x \in \mathcal{X}} g(x) \right)$ means sum of function $g(x)$ over all symbols x
- Example: For ABAAAABBAC,

  Length of encoded bit sequence $= \ell(\text{A}) n(\text{A}) + \ell(\text{B}) n(\text{B}) + \ell(\text{C}) n(\text{C})$

Goal

Goal: Prefix-free code with minimum length of encoded bit sequence

Formalizing the intuition

Fact

For any given sequence of symbols, the prefix-free code with minimum length of encoded bit sequence satisfies the following:
if $n(x) < n(x')$, then $\ell(x) \geq \ell(x')$

How to find such a code?

- For any given sequence of symbols, how to find the code with minimum length of encoded bit sequence?
  - **Huffman Codes**

"The problem of finding efficient prefix-free codes eluded researchers for quite some time; until David Huffman, then a mere graduate student in the EE dept at MIT solved it as his course project!"

Thank You!