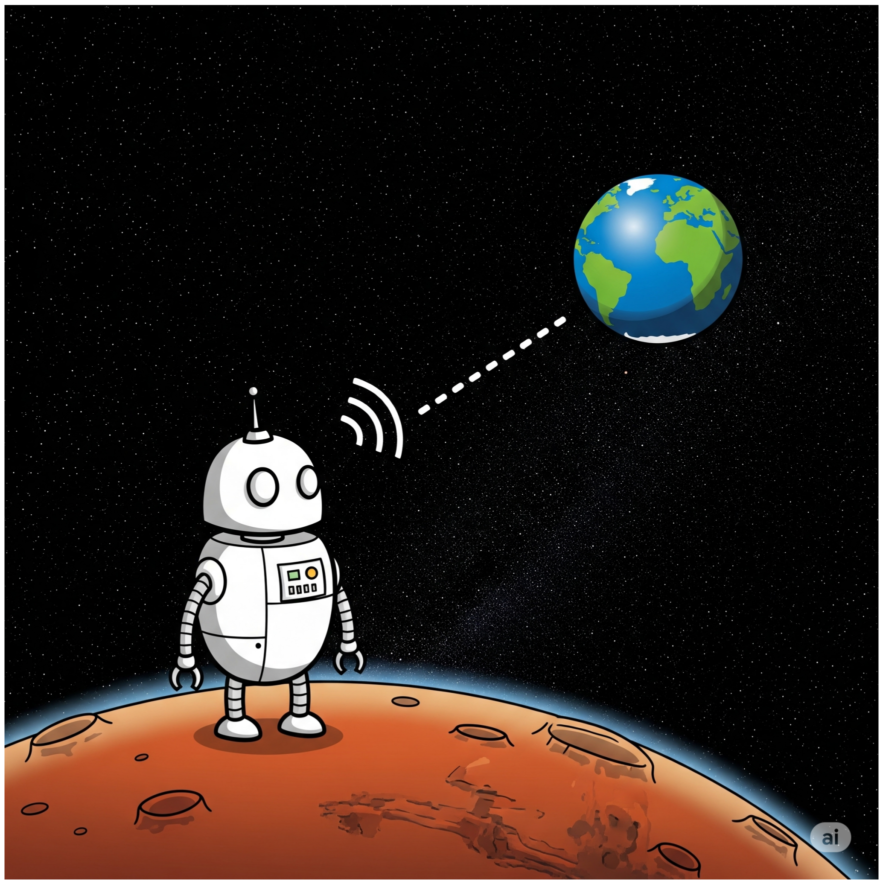
Message from Mars!

SIP 2025 Project



1. Introduction: Semantic Communication vs Classic Communication

Traditional communication systems aim to transmit data accurately by preserving every bit. These systems are designed around Shannon's theory, where communication success is measured by minimizing the bit error rate (BER). However, they are not aware of the meaning of the message.

Semantic communication, in contrast, prioritizes the accurate transmission of meaning, even if some bits are lost. The goal is to deliver messages that the receiver can understand—even if the exact words are missing. This shift is particularly important in low-resource, bandwidth-limited, or noisy environments.

Example: "Battery level is critical" could be received as "Battery critical" — the meaning is still preserved semantically, even if words are missing.

2. Problem Definition: Communication from Mars

Imagine a robot stationed on Mars that needs to send status updates to Earth mission control. Due to the nature of deep-space communication, the channel is:

- **Noisy** – prone to signal distortion and loss
- **Bandwidth-limited** – messages must be short or compressed
- **Latency-sensitive** – resending failed messages takes time

Your task is to simulate this scenario by implementing two different communication strategies and analyzing their effectiveness.

3. Systems to Implement

3.1. Classic Communication System

This system operates at the bit level and uses traditional techniques.

Components to implement:

1. Encode each message into binary
2. Apply three types of coding:
    - **One-hot indexing** (e.g., 4-bit index for 16 messages)
    - **Simple parity-based error detection/correction**
    - **Hamming code or custom fixed-length block code**
3. Add random bit-flip noise based on a simulated SNR
4. Decode and attempt to recover the original message
5. Calculate and plot Bit Error Rate (BER) across different SNR levels

3.2. Semantic Communication System

This system focuses on meaning preservation.

Steps:

1. Tokenize message and extract key words using:
    - `nltk` or `spaCy` for keyword extraction
2. Represent the message using word or sentence embeddings:
    - `spaCy` (https://spacy.io) for `.vector`

- `gensim` for Word2Vec embeddings

3. Simulate a noisy channel by randomly dropping words
4. Reconstruct the message using:
    - Cosine similarity (`sklearn.metrics.pairwise`)
    - BLEU score (`nltk.translate.bleu_score`)
5. Match to the closest original message

4. Noise and Decoding for Baseline

Simulate the channel by introducing noise:

- Use a random bit error generator (based on noise level)
- Simulate different SNR values (e.g., 0 to 10 dB)
- For each message and noise level:
    - Encode
    - Transmit through noisy channel
    - Decode
    - Measure and plot BER

5. Semantic Decoder and Evaluation

Recommended Tools

- `spaCy` with `en_core_web_md` or `lg` models:
  https://spacy.io/usage/models
- `gensim` Word2Vec or pretrained embeddings:
  https://radimrehurek.com/gensim/
- Cosine similarity function from `sklearn.metrics.pairwise`
- BLEU score from `nltk.translate.bleu_score`

Similarity Evaluation

- **Cosine Similarity:** Measures angle between two sentence vectors.
- **BLEU Score:** Compares n-gram overlap between original and decoded message.

Use these metrics to compare the decoded output to the original and report accuracy.

6. Comparison, Conclusion, and Future Steps

What to report:

- Comparison of BER vs. semantic similarity
- When does semantic communication outperform classic?
- Trade-offs between exactness and meaning

Possible extensions:

- Use transformer-based language models (e.g., BERT) to fill missing tokens
- Apply reinforcement learning to prioritize message words
- Test scalability with a larger message set or multilingual support