



Article

Document-Level Transformer-Based Text Semantic Communication System

Asma Mahgoub *  and Elias Yaacoub 

Department of Computer Science and Engineering, College of Engineering, Qatar University,
Doha P.O. Box 2713, Qatar; elias@qu.edu.qa

* Correspondence: am1510694@qu.edu.qa

Abstract

Given the emergence of new applications with stringent networking requirements, traditional bit-level communication may reach theoretical Shannon capacity and struggle to support such applications. Therefore, the idea of semantic communication (SC) has been proposed in literature. SC involves the use of artificial intelligence and a shared knowledge base to send a representation of data and reconstruct it at the receiver. In this paper, a SC system for document transmission is proposed. The system uses a transformer and a context encoder module to preserve the document's knowledge and context information by quantifying the relationship between a current sentence and the preceding sentences. The semantic information is sent over a noisy channel, and a mutual information maximization model is used to reduce the effect of noise on the transmitted signal. The proposed system is trained in two stages; the first stage involves training the sentence-level transformers while the second stage involves training the sentence-level and the context-level transformers. The performance of the system is evaluated using two publicly available datasets. The results show that the performance of the proposed system is better than a sentence-level SC model in terms of BLEU score and sentence similarity over a Rayleigh fading channel.

Keywords: deep learning; natural language processing; semantic communication; semantic similarity; text transmission

1. Introduction

Traditional transmission of text involves representing each character as bits and sending the encoded bits over the communication channel. This scheme has been used for many decades, but the existing network infrastructure may fail to support networking requirements for upcoming applications such as Metaverse and Internet of Everything. Hence, during the last five years, researchers have considered the use of semantic communication for the transmission of data. Semantic communication is a knowledge-based communication paradigm that involves the process of sending the meaning of data [1]. The sender transmits a representation of the message, which represents its meaning, and the receiver uses the received representation to infer the originally transmitted message with the aid of a shared knowledge base. The idea of semantic communication was initially introduced by Shannon and Weaver [2], as they have divided communication into three levels: technical, semantic and effective.

Different semantic communication systems were proposed in literature. As there is not yet a universal representation for different data modalities, semantic communication systems for different data types were proposed. Data types included text, images, audio



Academic Editor: Omneya A. Attallah

Received: 10 June 2026

Revised: 15 July 2026

Accepted: 17 July 2026

Published: 3 August 2026

Copyright: © 2026 by the authors.

Licensee MDPI, Basel, Switzerland.

This article is an open access article distributed under the terms and conditions of the [Creative Commons Attribution \(CC BY\) license](https://creativecommons.org/licenses/by/4.0/).

and video. In this paper, the focus will be on text transmission. Semantic transmission of text data is either done by transmitting embedding vectors or by transmitting relational triplets. It is noted that in most of the works, the focus is on transmitting a sentence without considering its context or its relationship with other sentences in a considered paragraph. Yet, to efficiently utilize the communicating resources, it is more meaningful to transmit a whole document rather than a single sentence. In this paper, a document-level text semantic communication system is proposed. The proposed system uses a context-aware document transformer [3] to semantically encode and decode documents. This transformer is utilized, because it does not depend on processing the whole paragraph when transmitting a single sentence nor does it focus on extensive token recursion during decoding. It is a simple window-based context-aware transformer.

The rest of the paper is structured as follows. Section 2 summarizes the previous work in the field of text semantic communication to provide a detailed description of the previously proposed models. Section 3 describes the proposed methodology. Section 4 presents the results. Finally, the paper is concluded in Section 5.

2. Literature Review

During the past years, many researchers have worked on developing semantic communication systems for text transmission. The aim of these systems is to represent and send the meaning of the text as either features or a set of relations. Commonly used techniques for building these systems were transformers, recurrent neural networks, reinforcement learning or other signal processing or machine learning techniques. The aim of these systems is to accurately reconstruct the transmitted message under different channel conditions.

Farsad, Rao and Goldsmith [4] proposed a joint source-channel coding for text transmission using deep learning. They illustrated that their proposed approach preserves the semantic information of the transmitted sentences as it is mapped into an embedding in the semantic space before being transmitted. Xie et al. proposed a deep learning enabled semantic communication system for text transmission (DeepSC) [5]. The proposed system uses a transformer to perform semantic encoding and decoding. A fully connected neural network is used to perform channel coding and decoding. Optimization is performed in terms of cross-entropy to predict the correct words in the received message and mutual information to overcome the channel noise. The system was compared to traditional communication schemes and was shown to have a better performance, particularly at low signal-to-noise (SNR) values. Reinforcement learning is used by Lu et al. [6] for semantic text transmission. The system is designed to optimize the semantic similarity between the sent and received messages. A Bi-directional LSTM model (Bi-LSTM) is used to encode the messages into their semantic representations. The reward functions used were the Bilingual Evaluation Understudy (BLEU) score and the CIDEr-D score. The results indicated the superiority of the proposed model in comparison to a cross-entropy-based model. Wang et al. [7] proposed a framework to enable a wireless base station to communicate with users using semantic communication. Semantic information is extracted from text by a knowledge graph. The text is represented as a set of relations and the most relevant triplets are transmitted over the network. At the receiver, graph-to-text algorithm is used to reconstruct the transmitted text. The proposed method was shown to reduce the data needed to be transmitted by the base station by 46%. Guo and Wang [8] designed a semantic communication system that optimizes semantic loss when transmitting a message. In their work, the focus is on designing a semantic communication system that optimizes the semantic loss -measured by a bidirectional encoder representations from a transformers (BERT) model when transmitting a message. Each message is represented as a vector using BERT. The proposed signal shaping method had lower semantic loss values

compared to binary phase shift keying (BPSK) and quadrature phase shift keying (QPSK). Zhou et al. [9] modify the DeepSC algorithm by adding an adaptive circulation mechanism to the transformer model to create a universal transformer which is more flexible with transmitting sentences with different semantic information. This modification resulted in a slight improvement in terms of BLEU scores and symbol error ratio (SER) compared to the original DeepSC algorithm. Liang et al. [10] propose a reasoning-based semantic communication system for text transmission. In their system, text is represented by a graph of entities and relations. The graph is embedded to a lower dimension and sent over the network. An algorithm is developed to infer lost information in the case of a packet loss. Life-long learning is used to update the knowledge base of the receiver. The proposed system was shown to achieve a 76% accuracy in inferring hidden information. Jiang et al. [11] proposed a system that transmits the important relations of an input text. The relations are extracted and sorted based on their semantic importance (or relevance) to the input text. The semantic importance was determined based on the semantic similarity between a given triplet and the transmitted text. A subset of the relations is transmitted depending on the channel's conditions. The proposed scheme was shown to be more reliable in preserving semantic information at low signal-to-noise ratio schemes; comparison was done between a traditional communication scheme that uses Huffman coding and LDPC channel coding and a deep learning-based joint source channel coding scheme. Liu et al. [12] propose a context-based semantic communication system. The system takes a paragraph as an input, extracts a current sentence and uses previous and following sentences as context information to enhance the semantic representation. The receiver uses the previously received sentences—i.e., the historical text—to decode the symbols received into the current sentence. The decoder uses a transformer-XL module. Performance is evaluated in terms of BLEU score, METEOR and BERT Similarity Score. Zhang et al. [13] developed a semantic communication system that uses part-of-speech, semantic similarity and context information. The proposed system decodes a received message into the originally sent sequence by maximizing the likelihood of observing an item given the previous N-1 items. To encode words based on their part-of-speech, Huffman tree was used. In each part-of-speech class, the words are sorted based on their probability of occurrence in the considered corpus. Semantically similar words are assigned to the same codeword. The tree is used in the decoding process. Dynamic programming is used to predict the received sequence by predicting word by word. The performance was compared with DeepSC and Huffman+LDPC scheme. Wang et al. [14] incorporate reinforcement learning into the system proposed in [7]. Semantic information is extracted from the input text by converting it into a set of relations. The base station uses reinforcement learning to optimize resource allocation and to determine the appropriate semantic information to be transmitted. The proposed approach reduces data transmission by 41%. Yao et al. [15] develop a semi-neural network-based model for semantic text transmission. Semantic encoding and decoding was done using two methods: Bi-LSTM and multi-head attention transformer. Traditional channel encoding and decoding schemes were used, namely, low density parity check (LDPC) codes. The system was optimized using cross-entropy and logistic loss. At the receiver side, iterative decoding is used along with a refinement encoder to reduce the semantic error between the transmitted and received message. The proposed method had a higher BLEU score compared to traditional communication schemes and to the full neural network-based scheme. Xu et al. [16] focus on using a knowledge base to enhance the performance of semantic text transmission. The proposed system uses transformers to perform semantic encoding and decoding. Semantic information transmission uses orthogonal frequency division multiplexing (OFDM). At the sender, entities extracted from sentence are used to obtain a relational triplet from the knowledge base. The triplet is sent as a pilot signal and

is used at the receiver. The triplet is passed through a knowledge encoder that enhances the output of the semantic decoder. A metric that measures knowledge-enhanced efficiency is proposed to illustrate the effect of the knowledge base on enhancing the performance of the system. The results demonstrate that the proposed method outperforms other methods in terms of BLEU score, BERT similarity score and word error rate (WER). Zhou et al. [17] focused on designing a semantic communication system that uses knowledge graphs. A message is represented as a relation triplet and encoded to be transmitted. At the receiver, an error correction algorithm is utilized to compare the received relation with the stored relations at the knowledge base and correct it. The triplet is converted to a message using T5, a text-to-text transformer model. The system was developed for a single-user scenario and a multi-user scenario. Also, an implementation using a software-defined radio was done to show the feasibility of the system. The performance was evaluated using BLEU and BERT similarity. Tang et al. [18] redesigned a semantic communication system for text transmission. The goal of their modified system is to preserve the sentence-level semantic fidelity rather than the word-level. To achieve this, the system used the same architecture as the DeepSC model, but the objective function was modified to include the semantic similarity metric calculated by the cosine similarity between the word embeddings. The results indicate that modifying the objective function enhanced the model's performance under an AWGN channel. Lee et al. [19] proposed an end-to-end semantic communication system that uses pretrained language models. Specifically, Bidirectional and Auto-Regressive Transformer (BART) encoder and decoder are used in the proposed system. To design a system compatible with 5G, polar codes are used in channel coding/decoding and modulation of 16-QAM is used. The use of BART pretrained weights made the system more robust to out-of-vocabulary words and reduced computational overhead. Performance is evaluated in terms of BLEU and Sentence BERT score. Chang, Hsu and Yang [20], used transformer architecture to develop a text semantic communication system. The decoder uses a recursive algorithm to generate lost or corrupted tokens with the aid of the previously decoded tokens and the tokens' relations extracted by the encoder. The partially decoded sentence is used in generating lost tokens. The method outperforms both DeepSC model [5] and the extended context-based method [12]. In [21], Wang et al. proposed the usage of large language models (LLMs) to enhance text semantic communication systems; the used language model was Vicuna. The model's tokenizer is used in the semantic encoding process and a pretrained LLM is used to estimate the token's probability during the decoding process. The proposed system is evaluated in terms of both semantic similarity and bit error rate. The performance was better than existing systems under an AWGN channel, but slightly worse under a Rayleigh channel.

It can be seen that the previous works mostly focus on transmitting a sentence and on preserving the semantic information within a sentence. Few works (such as [12,20]) focus on transmission of documents and preserving the semantic information within a whole document. But these works depended on a recursive decoding method [20], have a large number of parameters, need a large storage space, and are computationally extensive compared to other algorithms like DeepSC. Hence, the aim of this work is to design a simple document-level semantic communication system using a transformer model that exploits document-level context. The contributions of this work are summarized below:

- Designing a document-level semantic communication system that uses context information to preserve the semantic information of the whole document. The proposed system uses a document-level transformer instead of the standard transformer model.
- Investigating the effect of using a mutual information model on mitigating the effect of the channel's noise and on the system's performance.

- Studying the impact of using a shared embedding layer on the system's performance. The same embedding vectors are used for representing the words at the semantic encoder, decoder and context encoder.
- Illustrating the effect of training the system in two stages on the performance.
- Evaluating the proposed system's performance on transmission of news articles from the BBC articles dataset.
- Comparing the performance with a sentence-level semantic communication algorithm (DeepSC) and a traditional communication scheme that uses Huffman coding and Turbo codes.

3. Materials and Methods

In this paper, a semantic communication system for document transmission is developed. The proposed system integrates a document-level transformer model into the DeepSC system. The block diagram of the proposed system is shown in Figure 1. A document is represented as a set of sentences. Each sentence is sent through the transmission system after being encoded into a semantic representation using a sentence encoder and a context encoder. The context of a sentence is the previous two sentences. The receiver can encode the context locally to aid in decoding the received semantic representation. A channel encoder and decoder are utilized to overcome the noise of the communication channel. The semantic encoder consists of a context encoder and a sentence encoder. The architecture and the sublayers of the document-level transformer are presented in Figure 2. The semantic decoder is a sentence decoder that uses the context information (output of the context encoder) and semantic representation (output of sentence encoder) to extract the original sentence from the received semantic representation. It is worth noting that the semantic representation denoted by M is affected by channel noise before being fed into the sentence decoder. In this section, the blocks of the proposed system will be described in detail.

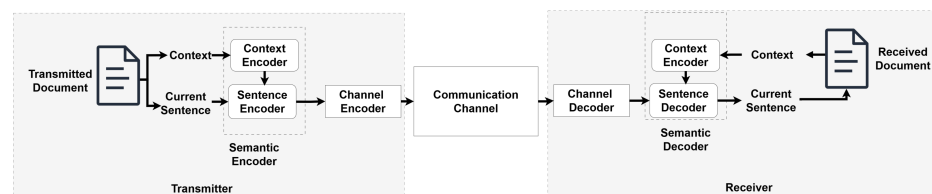


Figure 1. Block diagram of the proposed system.

3.1. Document-Level Machine Translation Algorithm

The architecture of the used transformer is inspired by the work of Zhang et al [3]. The authors proposed a document-level transformer model for machine translation by modifying the transformer architecture proposed by Vaswani et al. [22] to include the context information of the current sentence. Context was defined as the list of words from the two preceding sentences. Context information was extracted by utilizing a multi-head self-attention module to numerically represent the relationship between the context words and the sentence. The obtained vectors were then fed into a multi-head attention module at both encoder and decoder to integrate the context information into encoding and decoding processes. Their method outperformed the original transformer model in machine translation, and it had a small computational overhead. The simplicity of the proposed approach and the enhanced performance served as motivation to modify and extend it to a different application. Hence, by integrating this model with the DeepSC model a document-level semantic communication system is created.

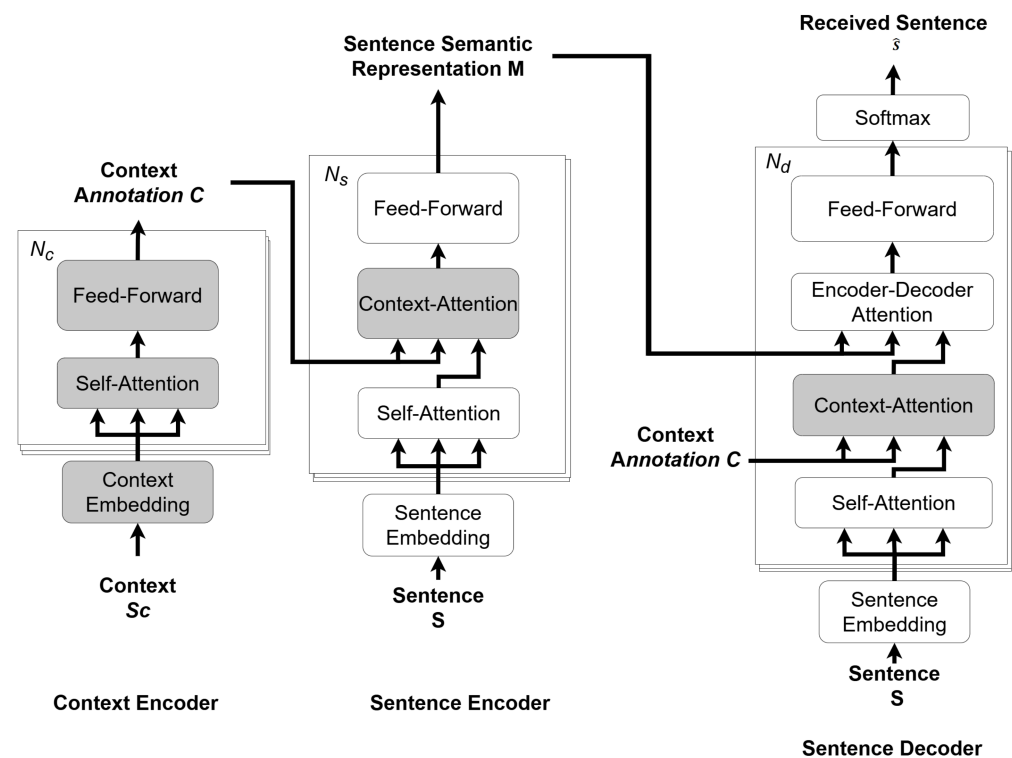


Figure 2. Detailed architecture of the transformers' modules. The context encoder has N_c layers, the sentence encoder has N_s layers, and the sentence decoder has N_d layers. The output of each layer is fed into the subsequent layer.

3.2. Integrating Document-Level Encoding with DeepSC

Xie et al. [5] developed a semantic communication system for text transmission. The proposed system uses the transformer model for semantic encoding. However, this model does not accurately capture the context of sentences and may not be suitable for document-level transmission. Hence, in this work, the DeepSC model is enhanced to include document-level semantic representation by integrating the document-level transformer with the semantic encoder and decoder of the DeepSC system. The system consists of a semantic encoder, channel layer and a semantic decoder, as shown in Figure 1. These will be described in the following subsections.

3.2.1. Semantic Encoder

The semantic encoder takes an input sentence and its context and encodes it into a semantic representation. The encoder consists of a context encoder and a sentence encoder as indicated in Figure 1.

Context Encoder

The context of a sentence S is denoted by S_C , a vector of all the α words in the previous j sentences. The number of preceding sentences j is referred to as the 'context length'. Each word is mapped into a vector representation by an embedding layer. The matrix S_C consists of a concatenation of all the α words in the previous sentences.

$$S_c = [w_1; w_2; \dots; w_\alpha] \quad (1)$$

where w_i is the vector representation of the word.

The architecture of the context encoder is identical to the traditional transformers' encoder. After passing the context via an embedding layer and a positional encoder, the context is fed into a multi-headed self-attention module. Then, the output is fed into

a feed-forward module. It is worth noting that there are residual connections after each sub-layer with layer normalization. Attention mechanism introduced in [22] is a function that maps a query vector Q to a key-value (K, V) pair. The query and keys have the same dimension d_k , the value vector has a dimension d_v . The query vector aims to find the important vectors—i.e., words—that affect it. It aims to find a score between a word and all the words in the sentence. Attention is calculated as the following:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \quad (2)$$

Multiheaded attention is done by linearly projecting the Q, K and V vectors h times and performing attention on all these projections. The attention values are then concatenated and linearly projected to produce a single output matrix. In this paper, multiheaded attention is denoted by the function $\text{MultiHead}(Q, K, V)$.

The multi-head self-attention is the process of calculating the multiheaded attention within a sentence (i.e., $Q = K = V$). The self-attention of the context matrix S_c is calculated as indicated in (3).

$$A = \text{MultiHead}(S_c, S_c, S_c) \quad (3)$$

After calculating the self-attention matrix, a layer normalization is applied with a residual connection as indicated in (4)

$$A = \text{LayerNorm}(A + S_c) \quad (4)$$

The layer normalization is applied on a data batch and is calculated using (5).

$$\text{LayerNorm}(\text{batch}) = \frac{\text{batch} - \mu_{\text{batch}}}{\sqrt{\sigma_{\text{batch}}^2 + \epsilon}} \cdot \gamma + \beta \quad (5)$$

where μ_{batch} is the batch's mean, σ_{batch}^2 is the batch's variance, ϵ is set to 10^{-5} and γ and β are learnable model parameters. Finally, the context annotation C is calculated using the column vector of each of the context words by feeding them into a position-wise feed-forward network $\text{FNN}(\cdot)$, as in (6).

$$C = [\text{FNN}(A_{.,1}); \dots; \text{FNN}(A_{.,\alpha})] \quad (6)$$

where $A_{.,i}$ is the column vector of the i_{th} context word. The context encoder has N_c layers. At each layer, Equations (3), (4) and (6) are computed using the output of the previous layer.

Sentence Encoder

The sentence encoder consists of cascaded N_s encoder layers; the output of each layer is the input to the subsequent layer and the output of the last layer (layer N_s) represents the semantic information of the sentence. The encoder follows a similar architecture to the context encoder, but an additional context-attention sub-layer is added. The current sentence denoted by S_0 is fed into an embedding layer to represent each word as a vector. The embedding vector denotes the word and its position. The sentence matrix is shown below in (7).

$$S_0 = [x_{start}; x_1; x_2; \dots; x_z; x_{end}] \quad (7)$$

where $\mathbf{x}_i$ is the vector representation of the i_{th} word in the current sentence. The vectors $\mathbf{x}_{start}$ and $\mathbf{x}_{end}$ represent the start and end tokens respectively. The sentence matrix is fed into a self-attention sub-layer. The output of the self-attention at layer n is denoted by:

$$\mathbf{B}_n = MultiHead(\mathbf{S}_{n-1}, \mathbf{S}_{n-1}, \mathbf{S}_{n-1}) \quad (8)$$

where $n \in \{1, \dots, N_s\}$ is the index of the encoder layer. At the first layer ($n = 1$), the input is the sentence $\mathbf{S}_0$ and at the subsequent layers the input is the output of the previous layer. The output of the attention layer is also combined with a residual connection and a layer normalization is applied (as in Equation (4)), but the equations are omitted for space limitations. The second sub-layer of the sentence encoder is the context multi-head attention. In this layer, the output of the context encoder $\mathbf{C}$ —shown in (6)—is fed into a multi-head attention module along with the output of the sentence's self-attention to obtain the matrix $\mathbf{D}^{(n)}$. Equation (9) is repeated for each encoder layer ($n \in \{1, \dots, N_s\}$)

$$\mathbf{D}^{(n)} = MultiHead(\mathbf{B}_n, \mathbf{C}, \mathbf{C}) \quad (9)$$

Finally, a position-wise feed-forward network is used to estimate the semantic representation at the n^{th} layer.

$$\mathbf{S}_n = \left[FNN\left(\mathbf{D}_{:,start}^{(n)}\right); FNN\left(\mathbf{D}_{:,1}^{(n)}\right); \dots; FNN\left(\mathbf{D}_{:,end}^{(n)}\right) \right] \quad (10)$$

where $\mathbf{D}_{:,i}^{(n)}$ is the column vector of the i_{th} word in the sentence at layer n . The output $\mathbf{S}_n \in \mathbb{R}^{E \times L}$ is the semantic representation of the current sentence at the n^{th} encoder layer. Where E is the embedding vector dimension and L is the length of the sentence. At each layer, Equations (8)–(10) are computed using the output of the previous layer. The semantic representation $\mathbf{M}$ of the considered sentence—shown in Figure 2—is computed at the last encoder layer N_s . Hence, $\mathbf{M} = \mathbf{S}_{N_s}$.

3.2.2. Channel Layer

To accurately design a semantic communication system, authors of the DeepSC [5] paper consider the presence of noise in the communication channel. To handle noise, a channel encoder and decoder modules were proposed. These modules are a set of fully-connected neural layers. At the channel encoder, the semantic representation $\mathbf{M}$ —which is equivalent to $\mathbf{S}_{N_s}$ obtained at the last layer of the sentence encoder—is encoded before passing through the noisy channel. The encoding process is denoted by (11)

$$\mathbf{X} = C_\beta(\mathbf{M}) \quad (11)$$

where C_β is the channel encoder. The encoder consists of two dense layers with 256 and 16 units, respectively. To transmit the features over the channel, the channel encoder was used to map the features to complex values. Particularly, the 16 output units indicate the use of 8 symbols per word, as each symbol is represented by a real and an imaginary number to be sent over the channel. The used activation function is ReLU (Rectified linear unit). After channel coding, the encoded signal $\mathbf{X}$ is transmitted over a noisy channel as illustrated in (12)

$$\mathbf{Y} = \mathbf{h}\mathbf{X} + \mathbf{n} \quad (12)$$

where $\mathbf{h}$ represents the Rayleigh fading channel coefficients. $\mathbf{h}$ is a matrix of complex coefficients representing the gain and phase shift of the transmitted signal. The matrix $\mathbf{h}$ has a complex gaussian distribution with mean 0 and covariance of 1 and $\mathbf{n}$ is the gaussian

noise with a mean of 0 and a covariance of σ^2 . In this implementation, $\mathbf{h}$ is a 2×2 matrix. This dimension represents two transmitting and two receiving antennas. At the receiver, the noisy received signal $\mathbf{Y}$ is passed through a channel decoder to estimate the received semantic representation $\hat{\mathbf{M}}$. The channel decoding process is denoted by (13).

$$\hat{\mathbf{M}} = \mathbf{C}_{\delta}^{-1}(\mathbf{Y}) \quad (13)$$

where $\mathbf{C}_{\delta}^{-1}$ is the channel decoder. The channel decoder consists of three dense neural layers with 128, 512 and 128 units respectively. The used activation function is ReLU and the output of the last layer is fed to a normalization layer. To overcome the effect of channel noise on the received signal, Xie et al. [5] utilize a mutual information (MI) model to maximize the MI between $\mathbf{X}$ and $\mathbf{Y}$. The MI between $\mathbf{X}$ and $\mathbf{Y}$ indicates how can knowing one variable affect the knowledge of the other. As discussed in [23], MI can be related to the Kullback–Leibler (KL) divergence between the marginal probability distributions $p(\mathbf{X})$ and $p(\mathbf{Y})$ and the joint probability distribution $p(\mathbf{X}, \mathbf{Y})$. Using a neural network, samples from $p(\mathbf{X})$, $p(\mathbf{Y})$, and $p(\mathbf{X}, \mathbf{Y})$ are drawn and used to estimate the MI loss as in (14).

$$\mathcal{L}_{MI} = \mathbb{E}_{p(\mathbf{X}, \mathbf{Y})}[\mathbf{f}_T] - \log\left(\mathbb{E}_{p(\mathbf{X})p(\mathbf{Y})}[\exp(\mathbf{f}_T)]\right) \quad (14)$$

where $\mathbf{f}_T$ is the output of a neural network estimated using samples from the corresponding distributions, i.e., from $p(\mathbf{X}, \mathbf{Y})$ for the first term and from $p(\mathbf{X})p(\mathbf{Y})$ for the second term. The used neural network has two hidden layers with 10 units. The expected value $\mathbb{E}$ is estimated by the arithmetic mean.

3.2.3. Semantic Decoder

The semantic decoder consists of a context encoder and a sentence decoder. The received semantic representation $\hat{\mathbf{M}}$ is fed into a sentence decoder to estimate the transmitted sentence $\hat{\mathbf{S}}$. The receiver utilizes the previously received sentences to calculate the context information locally by using the context encoder and feeds it into the sentence decoder. The decoder consists of N_d layers; the input of each layer is the output of the previous layer. Each layer consists of four sub-layers and there are residual connections between each sub-layer—as described in Equation (4)—but the equations are omitted for space limitations. The input of the decoder is the target output offset by one position. As the aim of the proposed system is to transmit the sentence, the input to the decoder will be the same as the input to encoder. This is represented as the following:

$$\mathbf{T}_0 = [\mathbf{x}_{start}; \mathbf{x}_1; \mathbf{x}_2; \dots; \mathbf{x}_z] \quad (15)$$

where $\mathbf{x}_{start}$ is the start-of-sentence token. The words and the start token will be represented by a vector representation that is obtained through an embedding layer. The matrix $\mathbf{T}_0$ is the concatenation of the vector representation (embedding) of each word. $\mathbf{T}_0$ is fed into a masked multi-head self-attention sub-layer. Masking is used to ensure that the decoder has access to the current word and the previously decoded words only. The self-attention of the n_{th} decoder layer is calculated as indicated below:

$$\mathbf{E}_n = \text{MultiHead}(\mathbf{T}_{n-1}, \mathbf{T}_{n-1}, \mathbf{T}_{n-1}) \quad (16)$$

where $n \in \{1, \dots, N_d\}$ and $\mathbf{T}_0$ is the shifted target sentence presented in (15). Then, the output of the attention layer will be fed into a context-attention sub-layer as in (17).

$$\mathbf{G}_n = \text{MultiHead}(\mathbf{E}_n, \mathbf{C}, \mathbf{C}) \quad (17)$$

where C is the output of the context encoder (6). After that, the output of the context-attention module will be fed into an encoder–decoder attention module

$$J^{(n)} = \text{MultiHead}(G_n, \hat{M}, \hat{M}) \quad (18)$$

where $\hat{M}$ is the semantic representation obtained at layer N_s that was estimated by the channel decoder at the receiver. Finally, a position-wise feed-forward neural network is used

$$T_n = [\text{FNN}(J_{:,1}^{(n)}); \dots; \text{FNN}(J_{:,z}^{(n)})] \quad (19)$$

where $J_{:,i}^{(n)}$ is the column vector of the i_{th} word in the sentence at layer n . The output $T_n \in \mathbb{R}^{E \times L}$ is the semantic representation of the current sentence at the n^{th} encoder layer. Where E is the embedding vector dimension and L is the length of the sentence. Note that C —used in calculating the multiheaded attention in (17)—is the context information that was estimated at the receiver side using the previously received sentences. However, to address any error propagation caused by previously received sentences, it is possible to receive this context information from the transmitter as well. The effect of using noisy context representation in sentence decoding will be discussed further in Section 4.5.

Finally, the received sentence $\hat{S}$ is estimated by predicting one word at a time using the matrix T_n and a SoftMax layer. Note that $n \in \{1, \dots, N_d\}$ for Equations (16)–(19) as each equation is applied on the current decoder layer. It is worth noting that the greedy-decode algorithm is used to predict each word in the received sentence based on the probability of each token from the vocabulary set.

3.2.4. Embedding Layer

As discussed earlier, there is an embedding layer to map a word into a vector representation. In the architecture presented in Figure 2, the embedding layer is different for the context-encoder, sentence-encoder and sentence-decoder. However, word embedding should give a consistent numerical representation of a word. Hence, it was thought that it would be better to use a shared set of weights among the sender and receiver when embedding words. In this work, the use of a separate embedding layer and shared embedding layer is studied to illustrate the effect of the embedding layer on the performance.

3.2.5. Loss Function

To optimize the weights of the proposed model, the cross-entropy loss is used to measure the difference between the received sentence $\hat{S}$ and the originally transmitted sentence S . The probability of the predicted word w_i is used to estimate the entropy and hence penalize the system based on how close or far the value is to 0. To measure the cross-entropy loss for a given predicted sentence, (20) is used.

$$\mathcal{L}(S, \hat{S}) = - \sum_{i=1}^z q(w_i) \log p(w_i) + (1 - q(w_i)) \log(1 - p(w_i)) \quad (20)$$

where $q(w_i)$ is the true probability that word w_i appears in the sentence at position i and $p(w_i)$ is the predicted probability. The loss function used in training the whole model combines $\mathcal{L}(S, \hat{S})$ and $\mathcal{L}_{MI}$ —presented in (14)—as indicated in (21)

$$\mathcal{L}_{total} = \mathcal{L}(S, \hat{S}) - \lambda \mathcal{L}_{MI} \quad (21)$$

the parameter λ was set to 0.0009 during the training.

3.3. Datasets

Two datasets were used to evaluate the performance of the proposed system. The first dataset is the BBC articles dataset (<https://www.kaggle.com/datasets/yufengdev/bbc-fulltext-and-category>, accessed on 4 January 2025) and the second dataset is the European Parliament proceedings dataset (<https://www.statmt.org/europarl/>, accessed on 26 April 2024) (Europarl) [24]. The BBC articles dataset contains around 2200 articles. For each article, a list of sentences was built. The sentences were preprocessed by converting them to ascii, removing numbers and punctuation and removing trailing spaces. Context information was stored for each article separately. The context of the current sentence was represented as the list of words in the preceding two sentences. The context length was set to 2 as it was indicated by Zhang et al. [3] that increasing the window further had no effect on performance and added a computational overhead. For each sentence, a start and end tokens were added, but the tokens were omitted from the context information. After extracting the sentences and the corresponding context from each article, the first 90% sentences were considered for the training set. The sentences were not shuffled and hence around 1997 articles were for training only and 228 articles were for testing only. This ensured that the context of an article was not leaked from training to testing. The Europarl dataset included proceedings of different parliament sessions in separate text files. In this study, the proceedings of each session was treated as a document. The preprocessing steps were the same as the BBC dataset. The dataset was converted to a list of sentences and a list of context information. Similarly, the context information considered the previous two sentences. The context information for the first sentence is an empty matrix and the context information for the second sentence is the list of words from the previous sentence. The size of the vocabulary along with the number of sentences and articles for each dataset are shown in Table 1.

Table 1. Size of the dataset.

Dataset	Size of Vocabulary	No. of Sentences	No. of Articles
BBC Articles	25,427	Training: 33,212 Test: 3690	Training: 1997 Test: 228
Europarl	23,094	Training: 93,470 Test: 10,385	Training: 9560 Test: 112

3.4. Training Modes

Training of the proposed system was done using two training modes [3]. Firstly, the model was trained with only the input and target sentences while omitting context data and context sub-layers (shaded blocks in Figure 2). This training mode was referred to as ‘sentence mode’. After training for half the number of training epochs, the context data is added, and the context encoder and context sub-layers are used and trained. Training and updating the weights is performed on encoder, decoder and context encoder. The weights of the modules trained in the first stage are updated as well during the second stage of training. This training mode is referred to as ‘context-mode’. Training the model using the two training modes is referred to as ‘transfer learning’ as using this training approach transfers the learned parameters of the sentence-based model into the context-based model. The algorithm for training the model using this approach is illustrated in Algorithm 1. The algorithm also illustrates the usage of MI maximization model in training. In this algorithm, $CR_{\theta}(\cdot)$ is the context encoder, $SR_{\omega}(\cdot)$ is the semantic sentence encoder and $SR_{\gamma}^{-1}(\cdot)$ is the semantic sentence decoder.

Algorithm 1: Training the Proposed Model

```

input : Number of epochs  $epochs$ , Batch size  $batch\_size$ , Sentence Dataset  $K$ ,
        Context Dataset  $K_c$ , MI Training flag  $MI$ 
output: Document-level Semantic Communication Model
         $CR_\theta(\cdot), SR_\omega(\cdot), C_\beta(\cdot), C_\delta^{-1}(\cdot), SR_\gamma^{-1}(\cdot)$ 
1  $batches = len(K)/batch\_size$ ;           // obtain number of batches in dataset
2  $context\_mode = False$ ;
3 for  $i = 1 \rightarrow epochs$  do
4   if  $i == epochs/2 + 1$  then           // Context-mode training
5      $context\_mode = True$ 
6   for  $i = 1 \rightarrow batches$  do
7     Transmitter:
8      $batchSource(K) \rightarrow S$ ;
9     if  $context\_mode$  then
10       $batchSource(K_c) \rightarrow S_c$ ;
11       $CR_\theta(S_c) \rightarrow C$ ;
12    else
13       $C = \emptyset$ ;                     // set to empty
14     $SR_\omega(S, C) \rightarrow M$ ;
15     $C_\beta(M) \rightarrow X$ ;
16    Transmit  $X$  over the channel;
17    Receiver:
18    Receive  $Y$ 
19     $C_\delta^{-1}(Y) \rightarrow \hat{M}$ ;
20    if  $context\_mode$  then           // Encode context locally
21       $batchSource(K_c) \rightarrow S_c$ ;
22       $CR_\theta(S_c) \rightarrow C$ ;
23    else
24       $C = \emptyset$ ;                     // set to empty
25     $SR_\gamma^{-1}(\hat{M}, C) \rightarrow \hat{S}$ ;
26    Compute loss function  $\mathcal{L}$  by (20).;
27    if  $MI$  then
28      Calculate  $P_{XY}, P_X$  and  $P_Y$ ;
29      Calculate MI loss using (14);
30      Calculate total loss  $\mathcal{L}_{total}$  using (21)
31    if  $context\_mode$  then
32      Train  $\omega, \beta, \delta, \gamma, \theta \rightarrow gradientDescent(\omega, \beta, \delta, \gamma, \theta)$ 
33    else                               // ignore context-based sublayers
34      Train  $\omega, \beta, \delta, \gamma \rightarrow gradientDescent(\omega, \beta, \delta, \gamma)$ 

```

4. Results

In this section, the proposed model is evaluated and compared with the original DeepSC model and a traditional communication scheme using different datasets. The model proposed by [5] (DeepSC) is highly cited and is used by many works as the standard model for semantic communication. In [25], the authors considered the use of DeepSC for semantic encoding/decoding and optimized their proposed multihop routing algorithm for maximum semantic spectral efficiency. DeepSC was also used when studying

resource allocation [26] in semantic-aware networks. In [27], the effect of radio frequency interference on DeepSC was studied. The continuous citation of DeepSC indicates that this model is popular and dominant in the field of text semantic communication and justifies the need to compare the proposed model with it. Other recently published context-aware models may have obtained better performance than DeepSC, but they were not yet adopted for implementing semantic communication systems in recent studies. Hence, besides a high performance, the proposed model has to have a moderate complexity to be implemented easily.

To assess the proposed model's performance and to determine the optimal model, several design considerations are tested and evaluated. The proposed system is evaluated under a Rayleigh fading channel. The considered traditional scheme uses turbo codes for channel coding and Huffman coding for source coding. Two modulation schemes were considered, namely, 64-QAM and 16-QAM. The turbo coding scheme had a code rate of 1/3 and the maximum number of decoding iterations was set to 4.

4.1. Implementation Details

The proposed system illustrated in Figure 1 has several hyperparameters. To determine the optimal hyperparameters, several values were tested and the set of hyperparameters that had the best performance was selected. The performance for different number of layers and different learning rate is discussed further in Section 4.5. Some hyperparameters like the size of the word embedding, the number of heads and the epochs were the same as [5]. In this work, the number of context encoder layers N_c was set to 1, and the number of encoder and decoder layers were both set to 3 ($N_s = N_d = 3$). The maximum length of the input sentences was set to 30. A summary of the hyperparameters and their values is shown in Table 2. A detailed summary of the model's architecture is presented in Table 3. The proposed model was trained and tested by Google Colab platform using NVIDIA Tesla T4 GPU. Training the model took around 37 min. The preprocessing, training and evaluation scripts for the proposed model are available online (<https://github.com/A-O-M55/DocumentDeepSC> (accessed on 15 January 2026)). It is worth noting that no random seed was fixed during training, but the performance was consistent across multiple runs. The model weights were initialized using Xavier uniform initialization.

Table 2. Hyperparameters of the proposed system.

Hyperparameter	Value
Maximum sentence length	30 (excluding start and end tokens)
Number of context encoder layers N_c	1
Number of encoder layers N_s	3
Number of decoder layers N_d	3
Size of word embedding	128
Number of heads	8
Learning rate	10^{-4}
MI learning rate	10^{-3}
Number of total training epochs	80
Batch size (training)	128
Optimizer	Adaptive moment estimation (ADAM)
Loss Function	Cross-entropy loss and MI loss (when MI model is used)

Table 3. Detailed architecture of the used semantic transmission system.

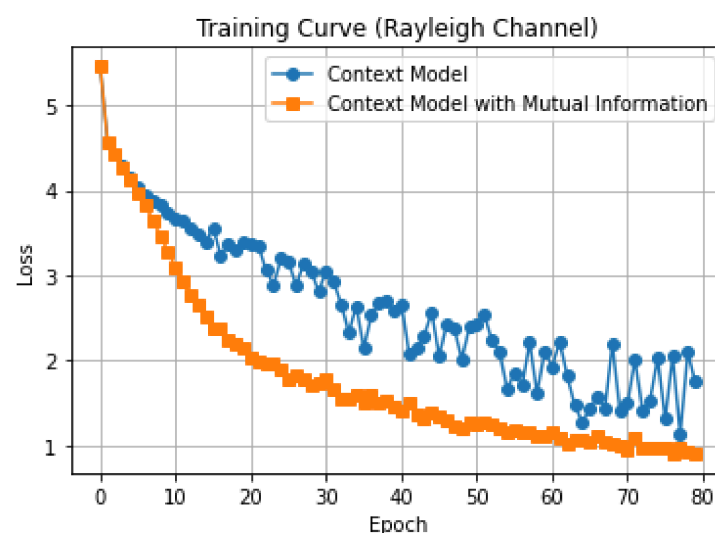
	Layer Name	Units	Activation
Encoder	1 × Context Encoder	128 (8 heads)	Linear
	3 × Transformer Encoder	128 (8 heads)	Linear
	Dense	256	Relu
	Dense	16	Relu
Channel	Rayleigh	None	None
Decoder	Dense	256	Relu
	Dense	128	Relu
	1 × Context Encoder	128 (8 heads)	Linear
	3 × Transformer Decoder	128 (8 heads)	Linear
	Prediction Layer	Dictionary Size	SoftMax
MI Model	Dense	10	Relu
	Dense	10	Relu
	Dense	1	Relu

4.2. Training Curves

To study the learning process of the proposed model, the cross-entropy loss of the sent and received sentence is plotted across different epochs. The effect of using an MI model and using transfer learning is illustrated in the following subsections.

4.2.1. Mutual Information

As stated earlier, the DeepSC system uses an MI model to maximize the MI between the sent and received signal and hence overcome the channel noise. To study the effect of using the MI model. Training of the proposed document-level system was done with the MI model and without it. Figure 3 shows the training curves under a Rayleigh fading channel. It is evident that the MI model has enhanced the training process as the loss values are lower when using the MI model, also the training curve is smoother and has fewer fluctuations. This is mainly due to ability of the MI model to reduce the effect of channel noise on the transmitted data.

**Figure 3.** Training curve with and without MI maximization model.

4.2.2. Transfer Learning

The proposed system was trained using two training modes. The training loss of a single training mode and transfer learning (two training modes) is shown in Figure 4. It

can be noted that at epoch 40, there is a jump in the loss value, this is due to the start of training the context-based model. Although the final loss value of both curves seems very close, the trained model converges more quickly when training is performed in two stages i.e., using transfer learning. Training using context-mode only takes around 42 min for 80 epochs. Whereas transfer learning takes 16 min for the first 40 epochs and 21 min for the last 40 epochs, meaning that it takes 37 min in total which is less by 10% of the original training time.

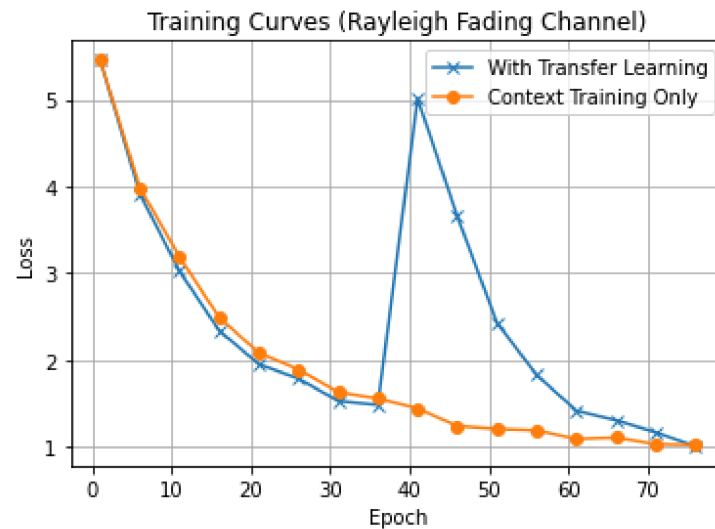


Figure 4. Training loss when training using context only (single-mode) vs. using transfer learning (sentence and context training).

4.3. Performance

The performance is quantified using the BLEU score and the BERT similarity score. The BLEU score is a metric used in machine translation; it calculates matching n-grams between a reference and a candidate translation [28]. The BLEU precision score is calculated by (22).

$$P_n = \frac{\sum_{ngram \in p} count_{match}(ngram)}{\sum_{ngram \in p} count(ngram)} \quad (22)$$

where $count_{match}$ counts number of matching n-grams between the reference sentence r (transmitted sentence) and the predicted sentence p (received sentence) and $count$ is the total n-grams in the predicted sentence. A penalty calculated using the length of the reference (l_r) and the predicted sentence (l_p) is defined in (23).

$$BP = \begin{cases} 1, & \text{if } l_p > l_r \\ \exp(1 - \frac{l_r}{l_p}), & \text{if } l_p \leq l_r \end{cases} \quad (23)$$

To calculate the BLEU score, (24) is used.

$$BLEU_N = BP \cdot \exp\left(\sum_{n=1}^N w_n \log P_n\right) \quad (24)$$

where $N = w_n = 1$ for calculating the 1-gram BLEU score. The BERT sentence similarity score is calculated using the cosine similarity as indicated in (25).

$$\text{BERT Score} = \frac{B(r) \cdot B(p)^T}{||B(r)|| ||B(p)||} \quad (25)$$

where $B(\cdot)$ is the BERT embeddings [29] of the given sentence. The reference sentence is r and the predicted sentence is p . In this work, the used BERT model is named “all-mpnet-base-v2”, which is based on a modified version of the BERT model named sentence BERT (SBERT) [30].

4.3.1. Performance on BBC Dataset

Several models were considered to estimate performance of the proposed system on the BBC articles’ dataset. The models are summarized below

1. Base Context-based Model: the original model described in Figures 1 and 2.
2. Base Model with MI: MI model is incorporated into the base model to maximize MI between sent and received signal.
3. Base Model with MI and Shared Embedding: same as the above model, but the embedding layer is shared among transmitter and receiver to map the same word to the same vector representation as described in Section 3.2.4: Embedding Layer.
4. Base Model with MI and Shared Embedding trained using transfer learning: same as the above model but training was done in two stages as described in Section 3.4.

The performance over a Rayleigh fading channel in terms of BLEU score and sentence similarity is summarized in Figures 5 and 6 respectively. The BLEU score is lower than 0.6 for all models when SNR is between 0 and 6 dB. The degraded performance at low SNR regimes indicates that there is a need to improve the channel encoding/decoding process to handle channel noise. For higher SNR values, the proposed model with transfer learning had a higher performance when compared to the DeepSC model. The hyperparameters for the considered DeepSC model were the same as the values listed in Table 2 except for the number of context encoder layers which is 0 for the DeepSC model. The proposed model preserves context of the transmitted documents and training in two stages enhanced the generalization capabilities of the model by transferring the knowledge learned during sentence-mode training to context-mode training which also avoids overfitting. It is noted that the performance was better than the base model when MI model was used, and it improved further when the embedding layer was fixed and with the introduction of transfer learning. A similar trend is observed when studying the performance in terms of sentence similarity. At low SNR values, the similarity is less than 0.6 and the model with a shared embedding layer had a slightly better performance than the DeepSC system. At higher SNR values, the model that is trained using transfer learning had a better performance compared to DeepSC. The “Turbo+Huffman” communication scheme with 16-QAM and 64-QAM had a low performance compared to all the considered models in terms of BLEU score. The sentence similarity scores for both “Turbo+Huffman” schemes were very low at an SNR of 0 to 12 dB. In comparison to traditional communication scheme, the proposed method had a better performance under a Rayleigh fading channel which is a complex and realistic channel model. Hence, semantic communication schemes are more suitable for handling complex channel conditions at low SNR values.

It is worth noting that a Rayleigh fading channel has a high probability of error for uncoded signals when the SNR is between 0 and 18 dB compared to other channels [31]. These relatively high error rates illustrate the challenging nature of this channel model and justifies the low performance for all schemes at a low SNR. Yet, it is important to note that these error rates refer to errors at the bit-level and should be distinguished from the semantic-based metrics reported in this study. In semantic communication, the goal is to preserve the semantic information. Even when the bits are distorted, the semantic decoder aims to find the most semantically similar word to the received distorted word. For instance, it is possible that two words have a similar meaning but a high bit error rate. The word ‘person’ and ‘individual’ refer to the same meaning, but are represented using

different characters and hence will have a high bit error rate. On the other hand, the words 'mat' and 'man' have a high character similarity although they are semantically different.

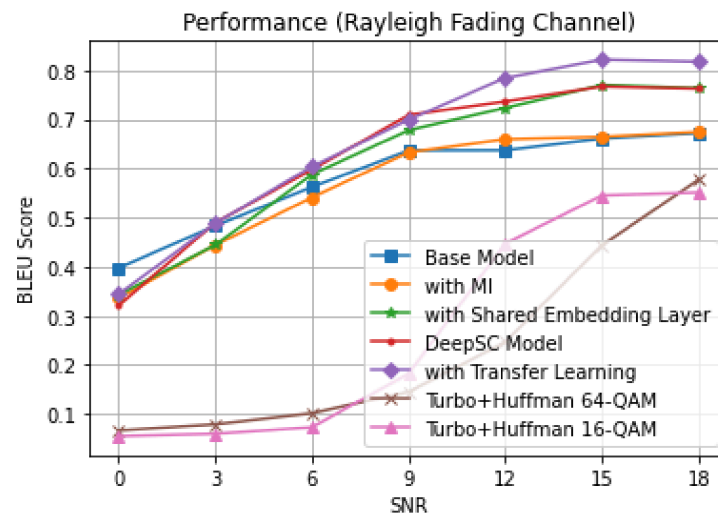


Figure 5. Performance of the proposed model in terms of BLEU over different SNR values.

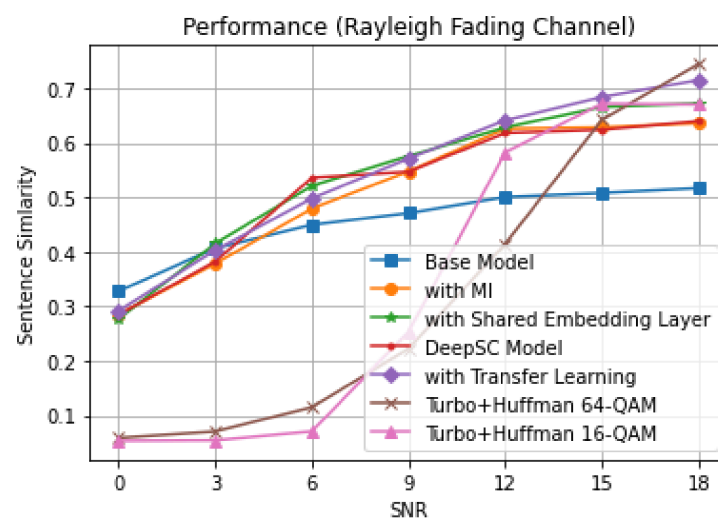


Figure 6. Performance of the proposed model in terms of sentence similarity over different SNR values.

4.3.2. Performance on Europarl Dataset

The BBC articles dataset is slightly challenging; it has a high number of unique vocabulary words but relatively few sentences. Hence, the proposed model was trained and tested on the Europarl dataset. The context model with transfer learning (model #4 explained in the previous section) was considered, because it had the highest performance over a Rayleigh fading channel. The same dataset was used to evaluate a DeepSC model with 3 encoder and decoder layers, with a maximum sentence length of 30 (excluding start and end tokens). The learning rate for the MI model of the DeepSC algorithm was set to 1×10^{-4} to prevent gradient explosion. The other hyperparameters were fixed to the default settings as indicated in the DeepSC paper [5]. The performance of a traditional communication scheme using turbo channel coding and Huffman source coding was evaluated on the Europarl dataset as well; the considered traditional scheme was modulated with 16-QAM and 64-QAM. The performance comparison in terms of BLEU and sentence similarity is shown in Figures 7 and 8 respectively. It can be noted that the performance is much better when compared to the results obtained using the BBC dataset (shown in

Figures 5 and 6). BLEU score and sentence similarity score had a value above 0.6 even at an SNR as low as 3 dB. The proposed system outperforms the DeepSC model and the traditional scheme at all SNR values.

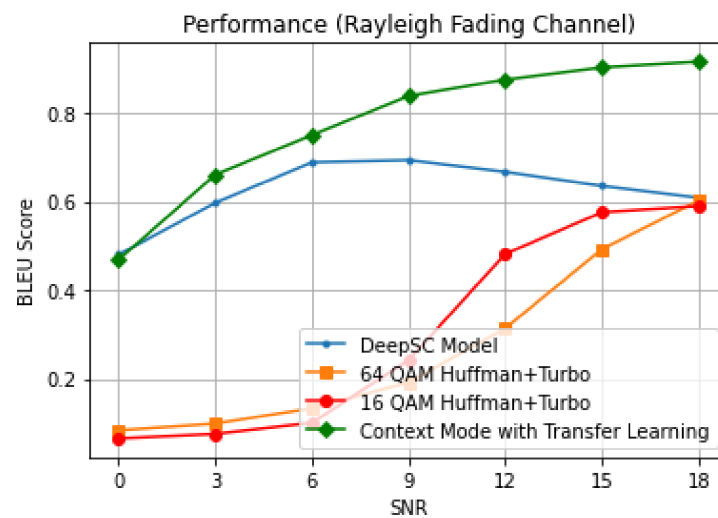


Figure 7. BLEU Score vs. SNR when Europarl Dataset is used.

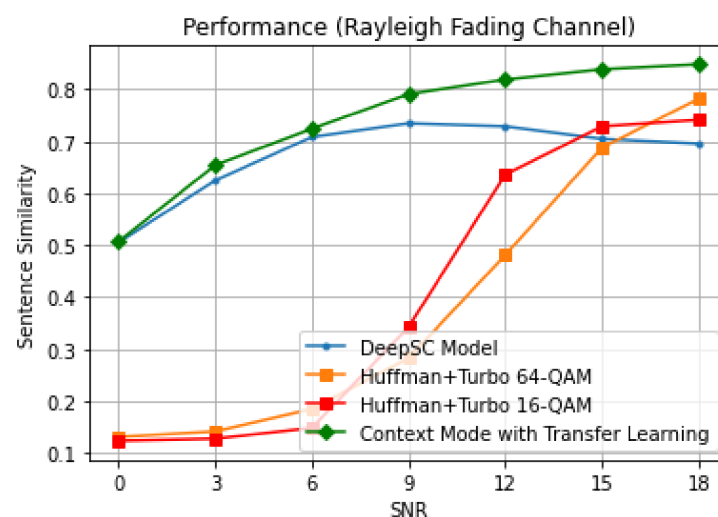


Figure 8. Sentence Similarity vs. SNR when Europarl dataset is used.

4.3.3. Performance Under Different Channel Models

As stated earlier, the proposed model was trained and tested under a Rayleigh channel. The Rayleigh channel model was considered for evaluation because it is more realistic compared to the other models. However, to illustrate the adaptability of the proposed model to other channels, the proposed model was evaluated under an AWGN channel and a Rician channel. The considered model is the model #4 described in Section 4.3.1 as it had the best performance. In an AWGN model, a signal is affected by gaussian noise, but no fading exist. While in a Rician channel, fading exists but there is a line of sight signal that makes the received signal stronger than in Rayleigh channels. The performance under different channels is shown in Figure 9. Although the considered model was trained under a Rayleigh channel, it had a higher performance under the other channels. The model under an AWGN channel had a slightly better performance than the Rician channel. The difference in performance is more evident at low SNR values. As the SNR increases beyond 15 dB, the gap in performance gets smaller.

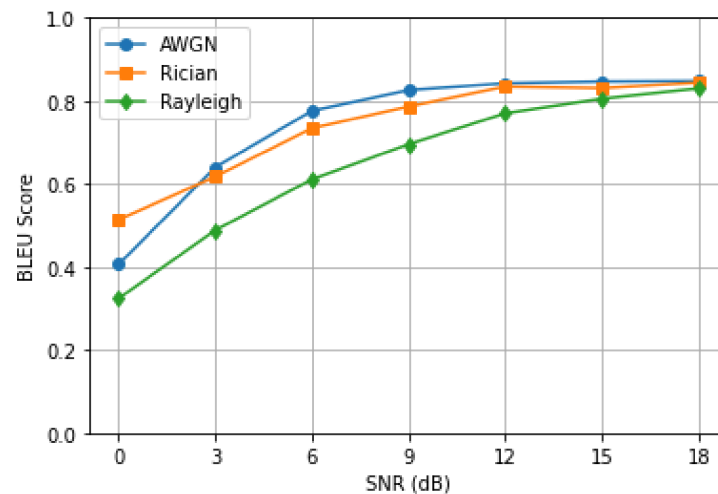


Figure 9. Comparison of the proposed model's performance under different channels.

4.3.4. Evaluation of Proposed Model Using Article Similarity Metrics

To study the performance of the proposed model on article similarity, the average BERT similarity score between the sent and received articles in the test set of the BBC dataset is considered. As indicated in Table 1, there are around 228 articles. To estimate article semantic similarity, the sentence similarity of the corresponding sentences in each article was calculated and accumulated. Then, the average semantic similarity for each article was obtained by dividing the total similarity score by the number of sentences. Equation (26) was used to calculate the article semantic similarity between the predicted article A_p and the reference article A_r .

$$\text{sim}(A_r, A_p) = \frac{1}{N_A} \sum_{i=1}^{N_A} \text{BERT_Score}(r_i, p_i) \quad (26)$$

where r_i and p_i are the i^{th} reference and predicted sentences in articles A_r and A_p respectively, N_A is the total number of sentences in the article, assuming that both A_r and A_p have the same number of sentences. The BERT similarity score is computed between the two sentences as shown in (25). The article similarity scores of the proposed model and other models are shown in Figure 10. To benchmark the proposed system, its performance was compared with another context-aware model, namely, GenSC model [20] along with the baseline DeepSC model and the traditional Turbo+Huffman communication scheme. The results show a similar pattern as the results of the previous subsection. However, it was noted that the values are less than the sentence similarity scores. Considering each article separately means that few sentences are considered to estimate the similarity score, this most likely contributed to the lower similarity scores. The proposed model is slightly better than the DeepSC model, particularly at an SNR above 15 dB. The 'Turbo+Huffman' scheme in 16-QAM shows a poor performance between 0 and 9 dB. The semantic communication-based schemes outperform the traditional scheme in preserving article's semantic similarity. The figure also shows that the GenSC model has a better performance compared to the proposed model. The performance of the proposed model could be improved further by using a higher learning rates, increasing the number of epochs or using masked language modeling for training the model. It is worth noting that GenSC has a higher performance as a trade-off in terms of having more parameters, a longer training time—which is almost $8 \times$ the training time of the proposed model—and a longer inference time, as it takes 7.14 ms to infer a single sentence. In this experiment, the GenSC model was trained for 80 epochs with a batch size of 64. The other hyperparameters for the GenSC model were the same

as the published model [20]. In the following subsection the complexity of the proposed model will be discussed to illustrate the difference in complexity between the proposed model and the GenSC model.

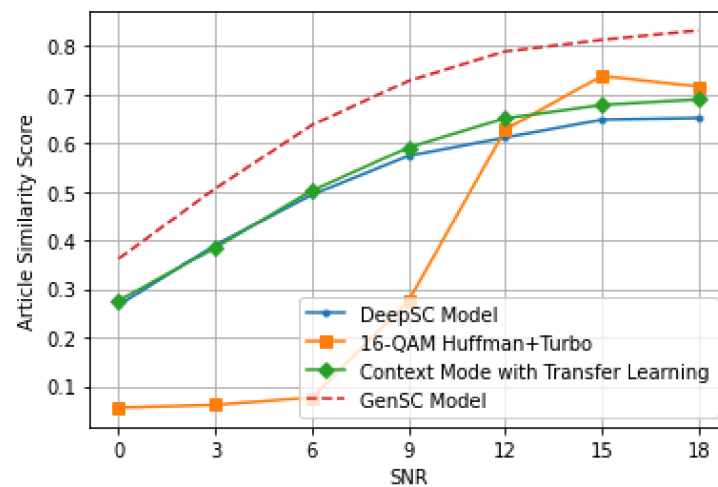


Figure 10. Performance under Rayleigh channel in terms of article similarity.

The article coherence is another document-based metric. This quantifies the logical flow of the sentences within an article. A document-based communication scheme should preserve the coherence within the received article. Article coherence is quantified by measuring the average semantic similarity between consecutive sentences [32]. The equation for article similarity is shown below.

$$Coherence(A_p) = \frac{1}{N_A - 1} \sum_{i=1}^{N_A-1} BERT_Score(s_i, s_{i+1}) \quad (27)$$

where A_p is the predicted article, N_A is the number of sentences in the article and s_i is the i^{th} sentence in article A_p . The article coherence is measured per article and is different based on the considered article. To obtain a comparison baseline, the ground truth value showing the average coherence score for all articles in the testing set is measured. Figure 11 shows the article coherence value for the proposed model at an SNR of 12 dB. Compared to the ground truth value, the semantic communication-based schemes preserved the coherence better than the traditional scheme. The proposed scheme preserved around 77% of the original article's coherence. It is worth noting that since coherence is measured per a single article, it is important to evaluate performance of the model based on both article similarity score and article coherence.

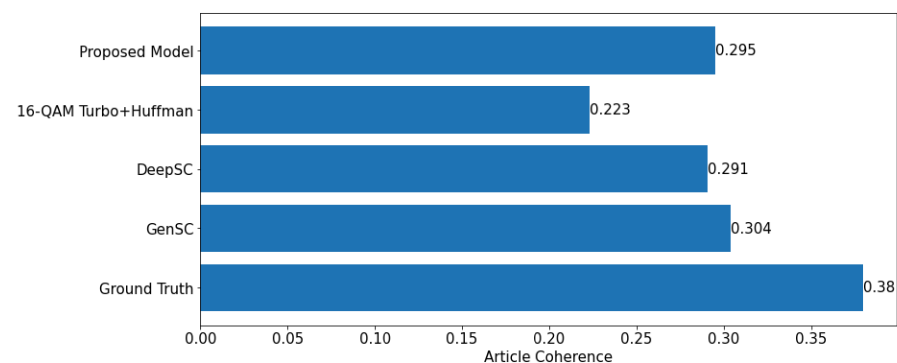


Figure 11. Average article coherence value of received articles at an SNR of 12 dB. Ground truth article coherence is shown to provide a reference.

As stated earlier, the proposed model uses previously received sentences to compute the context representation and decode the currently received sentence. To study the effect of error propagation on the received article's semantic similarity, the proposed model was evaluated under two scenarios. In the first scenario, the previous sentences were assumed to be received without any errors. The second scenario considers the errors of the previously received sentences. Figure 12 shows the effect of error propagation on the system's performance in terms of article semantic similarity. The system's performance decreases slightly as the error propagates from the previous sentences to the currently decoded sentences. To reduce the effect of error propagation on the performance, it is possible to consider implementing channel coding techniques or to use a channel encoder/decoder with a different architecture. This should be considered to enhance the overall performance and the performance at low SNR values.

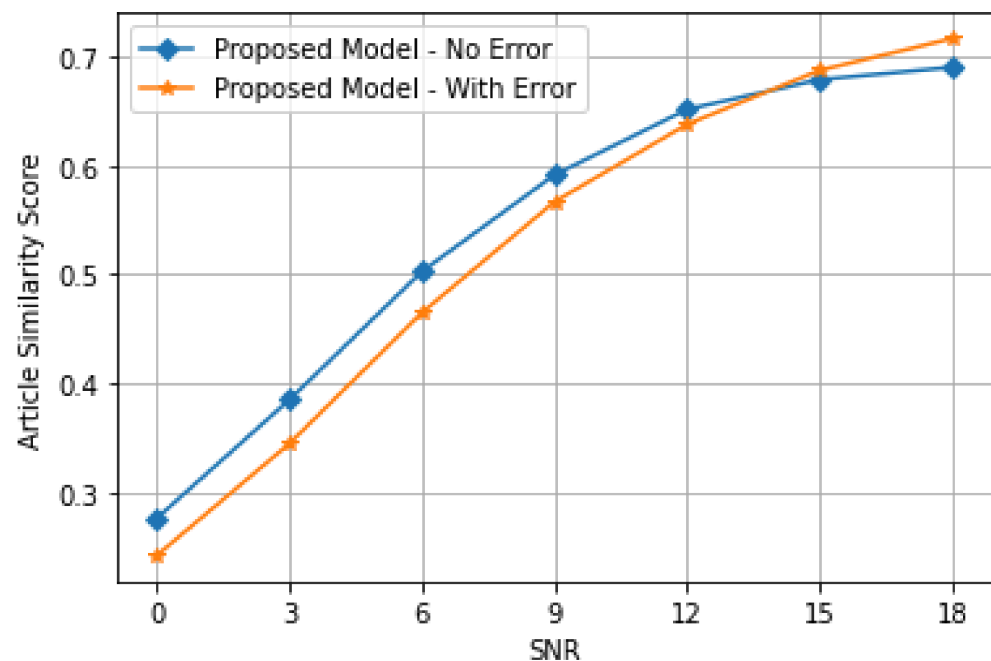


Figure 12. Effect of error propagation on performance in terms of article semantic similarity.

4.4. Complexity Analysis

The results in the previous section indicated that the document-level semantic communication system had outperformed the existing DeepSC model. However, this enhancement is due to adding an additional context encoder module and considering more sentences during the process of semantic encoding. Hence, it is crucial to study the added computational overhead of the proposed model. The considered model is the one that uses a shared embedding layer and MI maximization and is trained in two stages (model 4 described in Section 4.3). The batch size during inference was set to 64. Modifying the batch size did not affect the system's performance, but it slightly affected the inference time as low batch sizes showed slower inference times. However, increasing the batch size reduced inference time up to a saturation point. The inference time was measured per sentence, the proposed model processed a sentence along with its context in 3.5 ms which was slightly more than the inference time for processing a single sentence without its context by the DeepSC model. To evaluate the model's memory consumption during inference, the function 'torch.cuda_max_memory_allocated' was used. This function measures the maximum GPU memory occupied by tensors of an AI model. The peak memory for the proposed model is 20% more than the peak memory of the DeepSC model. This increase is expected due to the inclusion of additional context tokens in encoding and decoding. The other com-

plexity metrics of the models were computed using an evaluation metric script for pytorch (<https://github.com/xiamenwcy/Evaluation-Metrics-PyTorch/tree/main>, accessed on 3 June 2026). The pytorch version in this work is 2.3.0. The evaluation was done on a device with Intel Core i7-11800H CPU @ 2.30 GHz and NVIDIA GeForce RTX 3060 6 GB GPU. The results of the complexity analysis is summarized in Table 4. It can be noted that the proposed model has around 40% more parameters and hence a larger size, but the size is still relatively small and the model can be stored and executed in any end device and does not necessarily require storage in a cloud device. A device with 1 GB RAM can be used to run the model safely. It is worth noting that the number of parameters of the proposed model is less than the other existing context-based semantic communication models; it is less than the model proposed in [12] by 56% and less than [20] by 80%. This emphasizes the point that in comparison to other context-based models, the proposed model can integrate context without having a very large model size and without introducing a very high computational complexity. The GFLOPS of the proposed model are almost $1.6\times$ the GFLOPS of DeepSC. It is expected that processing more sentences per transmission would introduce a higher computational complexity. This is a trade-off between amount of processed data and computational complexity. Although the computational complexity of the proposed model is more, the required GFLOPS are moderate and the model can be executed locally or via an edge device in the network [33]. Tonini et al. [33] provide an analysis of the computing capabilities of nodes in an IoT network that utilizes goal-oriented communication. The analysis of the considered network nodes showed that an end device with a moderate computing power—such as a RaspberryPi 5—can support 31 GFLOPS. Emphasizing the idea that the minimum required GFLOPS for the proposed document-level transformer model is achievable by an end device in an IoT network.

Table 4. Complexity of the proposed document-level transformer and the DeepSC model.

	Document-Level Transformer	DeepSC
Inference Latency (ms)	3.51	2.86
Peak GPU Inference Memory per batch (MB)	213.8	175.8
Number of parameters	2.16 M	1.56 M
Size of model (MB)	8.22	5.95
Computational Complexity (GFLOPS)	10.11	6.28

4.5. Ablation Studies

In this section, several parameters will be altered to study their effect on the system's performance. Firstly, the number of symbols used to represent a word is investigated. This number represents the size of the output of the channel encoder layer as clarified in Section 3.2.2. The considered channel encoder maps each word to a vector of 16 values, as stated in the fourth row of Table 3. This number is $2\times$ No. of symbols which means that the number of symbols per word is 8. The performance of the proposed system was evaluated at an SNR of 9 dB to find the optimal number of symbols. The results are shown in Figure 13; increasing the number of symbols may make the system more robust to channel noise, but it also increases the complexity of the system by increasing the size of the channel encoder and decoder. The performance was optimal when 8 symbols were used; increasing the number of symbols further made the system more complex and did not have a significant improvement in the performance. It is worth noting that the maximum number of words per sentence is set to 30 and this means that the considered sentences are long and complex.

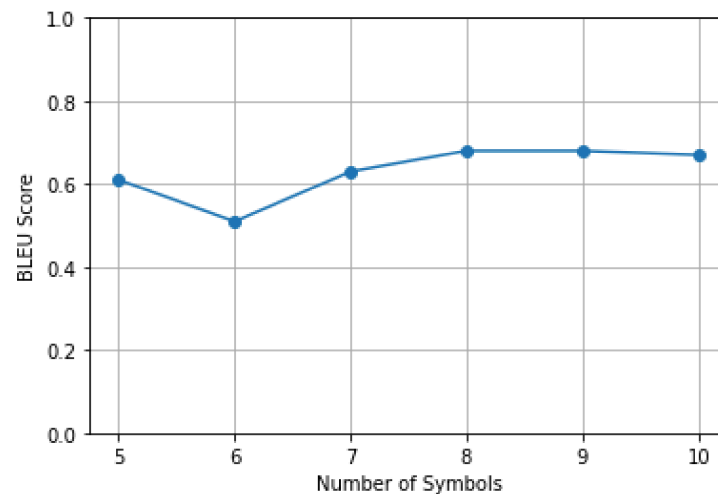


Figure 13. BLEU Score of the received sentences when different number of symbols are used to represent a word.

The effect of other hyperparameters on the system's performance was studied. Namely, the number of layers and the learning rate. The number of both encoder and decoder layers was set to 3 in the proposed system ($N_s = N_d = 3$), but other configurations were tested. A system using one, two and four layers was trained and tested. Figure 14 shows the performance for each configuration. It is worth noting that the number of layers indicates number of encoder layers and number of decoder layers separately and not combined. The results indicate that the general trend is that the performance improves when increasing the number of layers, but this is at a cost of an added complexity. Hence, to maintain a moderate complexity, the number of layers was set to 3. The learning rate controls the speed of updating the models' weights. The performance of the system trained under different learning rates is shown in Figure 15. A very low learning rate, such as 10^{-5} , will result in slow training and will lead to underfitting and hence a poor performance. On the other hand, a very high learning rate could result in gradient explosion and may also lead to overfitting. Hence, in this study the learning rate was set to 10^{-4} .

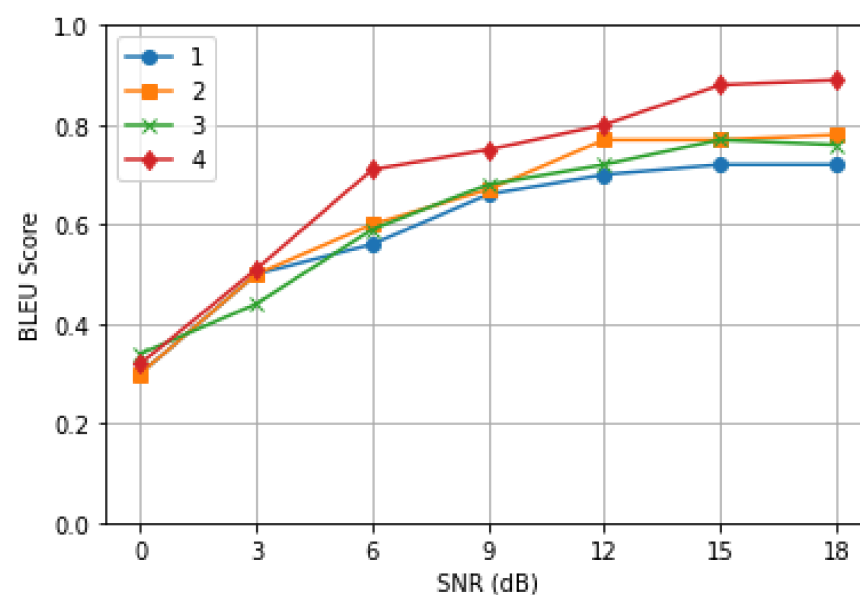


Figure 14. Performance of the proposed system when different number of layers are used.

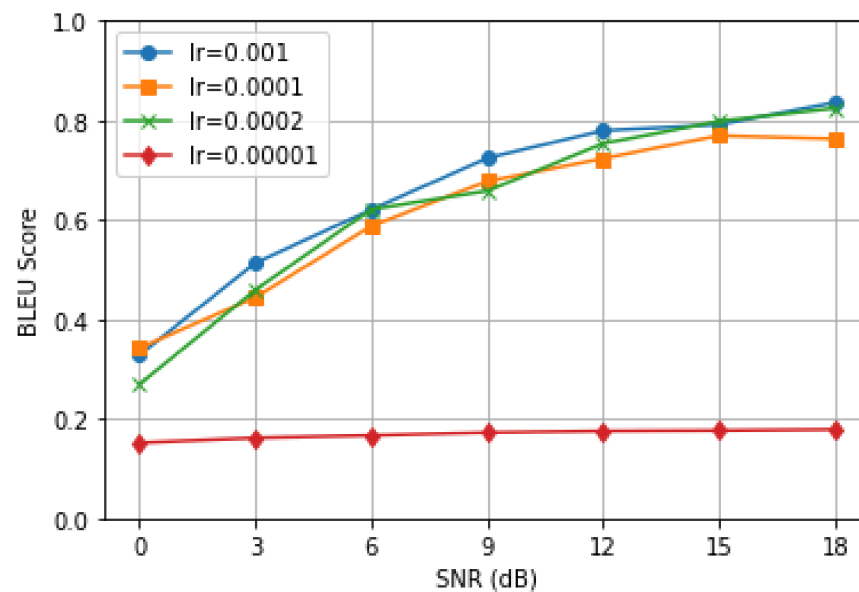


Figure 15. Performance of the proposed system when different learning rates are used during training.

In the proposed system, the transmitted sentence is encoded and decoded with the aid of the context information extracted from the previous sentences. The number of the considered sentences is referred to as the ‘context length’ that are used by the context encoder. To illustrate the effect of the context length, the proposed system was trained and tested when different context length values were used. The results are shown in Figure 16. It can be noted that there is no general trend for the effect of the context length on the system’s performance. At an SNR of 0 to 6 dB, the performance of a system with a context length of 4 is slightly better than using a length of 1 or 2. However, when the SNR is more than 6 dB, a context length of 2 achieves the optimal performance; this aligns with the findings of Zhang et al. [3] as they noted that increasing the context length more than 2 increases the complexity without significantly improving the performance. Hence, in this work, a context length of 2 sentences was considered. In the future, it is possible to investigate increasing the context length to enhance performance at low SNR values.

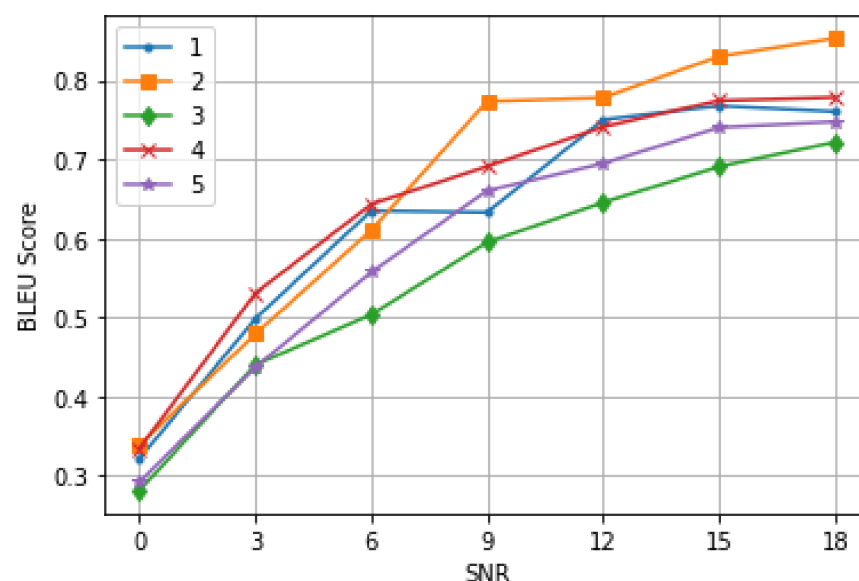


Figure 16. Effect of the context length on the system’s performance.

It is assumed in the proposed system that the context is encoded locally at the receiver. At the semantic decoder, the received symbols are decoded to obtain the received sentence with the locally encoded context information. However, it is possible to assume that both the current sentence and the context information are sent over the noisy channel and at the receiver, the semantic decoder uses the received information to predict the current transmitted sentence. It is important to study the effect of the channel noise on the system's performance when this approach is implemented. Figure 17 shows the performance when context information is sent over an ideal and a noisy channel. It is clear that the performance is almost identical in both cases with a slight deviation at an SNR of 9 dB and 18 dB. The identical performance indicates that the channel noise has a negligible effect on the performance as the system is trained to use joint source-channel coding to mitigate channel imperfections. The results imply that when the receiver has a limited computational power that prevents encoding the context locally, it is possible to get the encoded context directly from the transmitter without impacting the performance negatively. It is worth noting that the system is trained on the ideal channel scenario but tested on the noisy channel; this concept of “teacher-forcing” is introduced by [34] for training recurrent neural networks to prevent error propagation in training. The ground truth signal is used when training the network to predict future tokens. To address the issue of error propagation of previously received sentences, it is possible to utilize the received context representation when decoding the current received sentence.

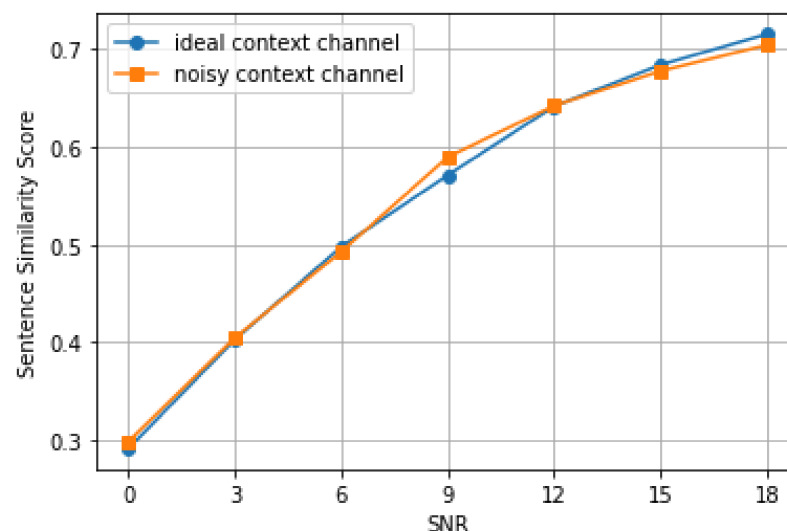


Figure 17. Effect of context noise on the performance of the system.

5. Conclusions

Semantic communication involves the transmission of the meaning of data instead of the actual bit representation. Semantic communication systems are currently designed to serve each data type separately. In this paper, the focus is on designing and evaluating a text semantic communication system. The proposed system handles transmission of documents and preserves the context information of the document by integrating a context encoder that uses previous sentences to semantically encode a current sentence. The system uses an MI maximization model to overcome the effect of channel noise. Also, a shared embedding layer is used to map each word to the same vector representation. The system is trained using transfer learning, where training is carried out in two stages: sentence-mode training and context-mode training. In the first stage, the context encoder module is ignored, and text is encoded and decoded using sentence encoders and decoders. In the second stage, the whole system—including the context encoder—is trained. The performance of the system

was evaluated using BLEU score and sentence similarity. Under a Rayleigh fading channel, the proposed model had a better performance in comparison to a sentence-level model, namely, the DeepSC model and a “Turbo+Huffman” communication scheme. Overall, the performance of the system was very low at low SNR values. This indicates the impact of channel noise on the transmitted message. It is possible to study error correction techniques or utilize dynamic allocation of the context window size and use a larger window at low SNR values. This may reduce the effect of channel and semantic noise on the transmitted sentence to enhance performance further. Additionally, the used dataset was slightly challenging as it had fewer sentences than the Europarl dataset, but more vocabulary words. This might be a cause for the poor generalization of the dataset. The system could be trained and tested on different articles’ datasets. To enhance this system and facilitate its adaptability, it is possible to implement a hardware prototype and test the feasibility of running the proposed algorithm in an edge device with moderate computational complexity. In this work, the considered Rayleigh channel involved two transmitting and receiving antennas. It is possible to consider different number of antennas by changing the size and dimension of matrix h to ensure the robustness of the proposed scheme. It is also possible to study performance on a network with interference or packet loss.

The results of this work are important to emphasize the importance of incorporating context information when transmitting documents using semantic communication techniques. Semantic communication of documents facilitates file sharing applications even with limited network resources. This would save network resources for more critical applications like healthcare and security.

Author Contributions: A.M.: Conceptualization of this study, Methodology, Software, Investigation, Validation, Writing—Original draft preparation. E.Y.: Writing—Review and Editing, Supervision. All authors have read and agreed to the published version of the manuscript.

Funding: This work was made possible by Qatar University–Student Grant no. QUST-CENG-2026-572. The research outcomes and statements made herein are solely the responsibility of the authors.

Data Availability Statement: The datasets used in this study are publicly available online. BBC dataset: <https://www.kaggle.com/datasets/yufengdev/bbc-fulltext-and-category>, accessed on 4 January 2025. Europarl dataset: <https://www.statmt.org/europarl/>, accessed on 26 April 2024.

Conflicts of Interest: The authors declare no conflicts of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript; or in the decision to publish the results.

Abbreviations

The following abbreviations are used in this manuscript:

SC	Semantic Communication
DeepSC	Deep learning enabled Semantic Communication
MI	Mutual Information
SNR	Signal-to-Noise Ratio
IoT	Internet of Things
BLEU	Bilingual Evaluation Understudy
BERT	Bidirectional Encoder Representations from Transformers
FLOPS	Floating-Points Operations per Second

References

1. Ren, J.; Sun, Y.; Du, H.; Yuan, W.; Wang, C.; Wang, X.; Zhou, Y.; Zhu, Z.; Wang, F.; Cui, S. Generative Semantic Communication: Architectures, Technologies, and Applications. *Engineering* **2026**, *56*, 45–61. [CrossRef]
2. Shannon, C.E.; Weaver, W. *The Mathematical Theory of Communication*; The University of Illinois Press: Champaign, IL, USA, 1949.

3. Zhang, J.; Luan, H.; Sun, M.; Zhai, F.F.; Xu, J.; Zhang, M.; Liu, Y. Improving the Transformer Translation Model with Document-Level Context. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, EMNLP 2018*; Association for Computational Linguistics: Brussels, Belgium, 2018; pp. 533–542. [\[CrossRef\]](#)
4. Farsad, N.; Rao, M.; Goldsmith, A. Deep Learning for Joint Source-Channel Coding of Text. In *Proceedings of the ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing-Proceedings*, Calgary, AB, Canada, 15–20 April 2018; pp. 2326–2330. [\[CrossRef\]](#)
5. Xie, H.; Qin, Z.; Li, G.Y.; Juang, B.H. Deep Learning Enabled Semantic Communication Systems. *IEEE Trans. Signal Process.* **2021**, *69*, 2663–2675. [\[CrossRef\]](#)
6. Lu, K.; Li, R.; Chen, X.; Zhao, Z.; Zhang, H. Reinforcement Learning-powered Semantic Communication via Semantic Similarity. *arXiv* **2021**, arXiv:2108.12121. [\[CrossRef\]](#)
7. Wang, Y.; Chen, M.; Saad, W.; Luo, T.; Cui, S.; Poor, H.V. Performance Optimization for Semantic Communications: An Attention-based Learning Approach. In *Proceedings of the 2021 IEEE Global Communications Conference (GLOBECOM)*; IEEE: Madrid, Spain, 2021; pp. 1–6. [\[CrossRef\]](#)
8. Guo, S.; Wang, Y.; Zhang, P. Signal Shaping for Semantic Communication Systems with A Few Message Candidates. In *Proceedings of the IEEE Vehicular Technology Conference*, London, UK, 26–29 September 2022. [\[CrossRef\]](#)
9. Zhou, Q.; Li, R.; Zhao, Z.; Peng, C.; Zhang, H. Semantic Communication with Adaptive Universal Transformer. *IEEE Wirel. Commun. Lett.* **2022**, *11*, 453–457. [\[CrossRef\]](#)
10. Liang, J.; Xiao, Y.; Li, Y.; Shi, G.; Bennis, M. Life-long Learning for Reasoning-based Semantic Communication. In *Proceedings of the 2022 IEEE International Conference on Communications Workshops, ICC Workshops 2022*, Seoul, Republic of Korea, 16–20 May 2022; pp. 271–276. [\[CrossRef\]](#)
11. Jiang, S.; Liu, Y.; Zhang, Y.; Luo, P.; Cao, K.; Xiong, J.; Zhao, H.; Wei, J. Reliable Semantic Communication System Enabled by Knowledge Graph. *Entropy* **2022**, *24*, 846. [\[CrossRef\]](#) [\[PubMed\]](#)
12. Liu, Y.; Jiang, S.; Zhang, Y.; Cao, K.; Zhou, L.; Seet, B.C.; Zhao, H.; Wei, J. Extended context-based semantic communication system for text transmission. *Digit. Commun. Netw.* **2022**, *10*, 568–576. [\[CrossRef\]](#)
13. Zhang, Y.; Zhao, H.; Wei, J.; Zhang, J.; Flanagan, M.F.; Xiong, J. Context-Based Semantic Communication via Dynamic Programming. *IEEE Trans. Cogn. Commun. Netw.* **2022**, *8*, 1453–1467. [\[CrossRef\]](#)
14. Wang, Y.; Chen, M.; Luo, T.; Saad, W.; Niyato, D.; Poor, H.V.; Cui, S. Performance Optimization for Semantic Communications: An Attention-Based Reinforcement Learning Approach. *IEEE J. Sel. Areas Commun.* **2022**, *40*, 2598–2613. [\[CrossRef\]](#)
15. Yao, S.; Niu, K.; Wang, S.; Dai, J. Semantic Coding for Text Transmission: An Iterative Design. *IEEE Trans. Cogn. Commun. Netw.* **2022**, *8*, 1594–1603. [\[CrossRef\]](#)
16. Xu, X.; Xiong, H.; Wang, Y.; Che, Y.; Han, S.; Wang, B.; Zhang, P. Knowledge-enhanced semantic communication system with OFDM transmissions. *Sci. China Inf. Sci.* **2023**, *66*, 172302. [\[CrossRef\]](#)
17. Zhou, F.; Li, Y.; Zhang, X.; Wu, Q.; Lei, X.; Hu, R.Q. Cognitive Semantic Communication Systems Driven by Knowledge Graph. In *Proceedings of the IEEE International Conference on Communications*, Seoul, Republic of Korea, 16–20 May 2022; pp. 4860–4865. [\[CrossRef\]](#)
18. Tang, B.; Li, Q.; Huang, L.; Yin, Y. Text Semantic Communication Systems with Sentence-Level Semantic Fidelity. In *Proceedings of the IEEE Wireless Communications and Networking Conference, WCNC*, Scotland, UK, 26–29 March 2023. [\[CrossRef\]](#)
19. Lee, J.H.; Lee, D.H.; Sheen, E.; Choi, T.; Pujara, J. SEQ2SEQ-SC: End-To-End Semantic Communication Systems with Pre-Trained Language Model. In *Proceedings of the Conference Record-Asilomar Conference on Signals, Systems and Computers*, Pacific Grove, CA, USA, 29 October–1 November 2023; pp. 260–264. [\[CrossRef\]](#)
20. Chang, M.K.; Hsu, C.T.; Yang, G.C. GenSC: Generative Semantic Communication Systems Using BART-Like Model. *IEEE Commun. Lett.* **2024**, *28*, 2298–2302. [\[CrossRef\]](#)
21. Wang, Z.; Zou, L.; Wei, S.; Li, K.; Liao, F.; Mi, H.; Lai, R. Large-Language-Model-Enabled Text Semantic Communication Systems. *Appl. Sci.* **2025**, *15*, 7227. [\[CrossRef\]](#)
22. Vaswani, A.; Brain, G.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Łukasz Kaiser; Polosukhin, I. Attention is All you Need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems; (NIPS 2017)*; Curran Associates Inc.: Red Hook, NY, USA, 2017; pp. 6000–6010.
23. Belghazi, M.I.; Baratin, A.; Rajeswar, S.; Ozair, S.; Bengio, Y.; Courville, A.; Hjelm, D. Mutual Information Neural Estimation. In *Proceedings of the 35th International Conference on Machine Learning*; PMLR: Stockholm, Sweden, 2018; pp. 531–540.
24. Koehn, P. Europarl: A Parallel Corpus for Statistical Machine Translation. *Proceedings of Machine Translation Summit X: Papers*, Phuket, Thailand, 13–15 September 2005.
25. Saad, M.; Abdallah, S.; Dahrouj, H. Multihop Routing for Maximum Semantic Spectral Efficiency. *IEEE Wirel. Commun. Lett.* **2026**, *15*, 2473–2477. [\[CrossRef\]](#)
26. Yan, L.; Qin, Z.; Zhang, R.; Li, Y.; Li, G.Y. Resource Allocation for Text Semantic Communications. *IEEE Wirel. Commun. Lett.* **2022**, *11*, 1394–1398. [\[CrossRef\]](#)

27. Getu, T.M.; Saad, W.; Kaddoum, G.; Bennis, M. Performance Limits of a Deep Learning-Enabled Text Semantic Communication Under Interference. *IEEE Trans. Wirel. Commun.* **2024**, *23*, 10213–10228. [\[CrossRef\]](#)
28. Papineni, K.; Roukos, S.; Ward, T.; Zhu, W.J. BLEU: A Method for Automatic Evaluation of Machine Translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics-ACL '02*; Association for Computational Linguistics: Philadelphia, PA, USA, 2001; p. 311. [\[CrossRef\]](#)
29. Devlin, J.; Chang, M.W.; Lee, K.; Toutanova, K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the NAACL HLT 2019-2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies-Proceedings of the Conference, Minneapolis, MN, USA, 2–7 June 2018*; Volume 1, pp. 4171–4186.
30. Reimers, N.; Gurevych, I. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*; Association for Computational Linguistics: Hong Kong, China, 2019; pp. 3980–3990. [\[CrossRef\]](#)
31. Shankar, P.M. *Fading and Shadowing in Wireless Systems*, 2nd ed.; Springer International Publishing: Berlin/Heidelberg, Germany, 2017; pp. 1–464. [\[CrossRef\]](#)
32. de Oliveira Braz Junior, O.; Fileto, R. Semantic Coherence of Short Text at the Word Level. *J. Braz. Comput. Soc.* **2025**, *31*, 449–464. [\[CrossRef\]](#)
33. Tonini, F.; Lanci, P.; Borsatti, D.; Poe, W.Y.; Trivisonno, R.; Cerroni, W. End-to-End Performance Analysis for Intelligent IoT Devices in Goal-Oriented Networking. In *Proceedings of the 2025 IEEE 11th International Conference on Network Softwarization (NetSoft)*; IEEE: Budapest, Hungary, 2025; pp. 1–6. [\[CrossRef\]](#)
34. Williams, R.J.; Zipser, D. A Learning Algorithm for Continually Running Fully Recurrent Neural Networks. *Neural Comput.* **1989**, *1*, 270–280. [\[CrossRef\]](#)

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.