Politecnico di Torino

Politecnico di Torino

Computer Engineering LM-32

Academic Year 2024/2025

Graduation Session December 2025

Knowledge Graph-Guided and LLM-Based Semantic Communication for Challenging Edge Networks

Supervisors:

Alessio Sacco

Guido Marchetto

Candidate:

Ten. Loris Bacaloni

Abstract

Traditional wireless communication systems, based on Shannon's information theory, are designed to ensure that every transmitted bit is received exactly as sent. This bit-level accuracy is efficient when channels are stable and bandwidth is enough, but it becomes inefficient in real environments where noise, fading, interference, or low bandwidth can distort signals. In many modern applications, such as Internet of Things (IoT) sensors, unmanned aerial vehicles (UAVs), or edge computing nodes, what truly matters is the meaning of the message.

Semantic communication aligns perfectly with this shift in focus, which transitions from bit accuracy to message preservation. Instead of sending every word or symbol, the system seeks to transmit an encoded compact representation of the underlying meaning, which can then be reconstructed at the receiver using shared models of language and knowledge.

Our work proposes an end-to-end semantic communication framework for text that integrates two complementary technologies. The first is the Knowledge Graph (KG), a structured network that represents entities and the relationships between them, capturing the essential semantic structure. The second component is a Large Language Model (LLM), trained to understand and generate natural language and capable of encoding and reconstructing the semantics of a message. KGs offer structured semantic grounding while LLMs handle contextual encoding and decoding, enabling efficient meaning transmission under bandwidth or noise limits.

At the transmitter, a natural-language processing pipeline, based on spaCy (a widely used industrial NLP toolkit) and OpenIE (Open Information Extraction), analyzes input sentences to extract their main entities, relations, and summary statements. The result is a set of triples that form a Knowledge Graph representation. A sequence-to-sequence (seq2seq) encoder, such as T5 or BART, then performs semantic compression, transforming the text into a compact sequence of tokens and their contextual embeddings.

This encoded representation is sent over a wireless channel affected by typical physical-layer damages such as fading (signal weakening due to movement or obstacles), multipath propagation (the signal taking multiple paths and arriving at different times), additive white Gaussian noise (AWGN), and interference from other devices. These defects can corrupt the transmitted sequence, challenging the receiver to recover meaning despite distortion.

On the receiver side, a two-phase semantic decoder reconstructs the message. First, the same LLM used at the transmitter tries to rebuild the original message by predicting the most likely words from the received, possibly corrupted, tokens

or embeddings. Then, a BERT-based masked-language model refines the output, validating and correcting uncertain words using contextual reasoning.

We evaluate our system on a sentiment analysis dataset (SST-2) across different signal-to-noise ratio (SNR) levels, testing both the quality of semantic reconstruction and its robustness to transmission errors. We also perform ablation studies to assess the KG's impact, transmission mode, and decoder configuration. The results show that this KG-LLM hybrid framework reduces transmitted data size while maintaining high semantic fidelity under channel deterioration. This shows that transmitting meaning, rather than raw bits, can achieve a more efficient, resilient, and context-aware communication process.

Table of Contents

List of Tables

List of Figures

Glossary

AMR

Abstract Meaning Representation

AWGN

Additive White Gaussian Noise

BART

Bidirectional and Auto-Regressive Transformer

BERT

Bidirectional Encoder Representations from Transformers

BERTScore

BERTScore similarity metric

BER

Bit Error Rate

BLEU

Bilingual Evaluation Understudy

DL

Deep Learning

FEC

Forward Error Correction

IoT

Internet of Things

KG

Knowledge Graph

LLM

Large Language Model

MLM

Masked Language Modelling

METEOR

Metric for Evaluation of Translation with Explicit Ordering

MSE

Mean Squared Error

NER

Named Entity Recognition

NLP

Natural Language Processing

OIE

Open Information Extraction

OWL

Web Ontology Language

POS

Part-of-Speech tagging

QAM

Quadrature Amplitude Modulation

QPSK

Quadrature Phase Shift Keying

RAG

Retrieval-Augmented Generation

RDF

Resource Description Framework

ROUGE-L

ROUGE-L text similarity metric

SBERT

Sentence-BERT model

SER

Symbol Error Rate

SNR

Signal-to-Noise Ratio

SPO

Subject-Predicate-Object triple

SST-2

Stanford Sentiment Treebank v2 dataset

T5

Text-to-Text Transfer Transformer

TER

Token Error Rate

UAV

Unmanned Aerial Vehicle

Chapter 1

Introduction

1.1 Background and Motivation

The exponential increase in connected devices and the diffusion of edge computing have deeply transformed the communication landscape. Modern networks must support a growing number of heterogeneous nodes, from IoT sensors to autonomous aerial vehicles, that operate under strict constraints of bandwidth, latency, and energy consumption.

In these contexts, the traditional approach to data transmission, grounded in Shannon's information theory, reveals intrinsic limitations. Its objective of ensuring bit-level accuracy, guaranteeing that each bit received matches the transmitted one, is efficient only in ideal conditions. When the wireless channel is affected by noise, fading, or interference, the cost of preserving perfect fidelity becomes prohibitive and misaligned with the true purpose of communication, the transmission of meaning.

This observation recalls the theoretical foundation established by Claude Shannon and Warren Weaver in their seminal work, *The Mathematical Theory of Communication* [1]. Their model, as depicted in *Figure 1.1*, formalized the communication process as a linear chain composed of an *information source*, a *transmitter* (or encoder), a *channel* affected by *noise*, a *receiver* (or decoder), and a *destination*. At its core, Shannon's formulation introduced a quantitative measure of information, *entropy*, and defined the goal of communication as the accurate transmission of symbols through a noisy medium. Weaver later articulated that communication can be viewed at **three distinct levels**: the *technical* problem of symbol accuracy, the *semantic* problem of meaning, and the *effectiveness* or *pragmatic* problem concerning the influence of messages on behavior. However, the mathematical theory itself deliberately addressed only the first level, focusing on the engineering challenge of reproducing signals faithfully, while explicitly excluding semantics as "irrelevant to the engineering problem."

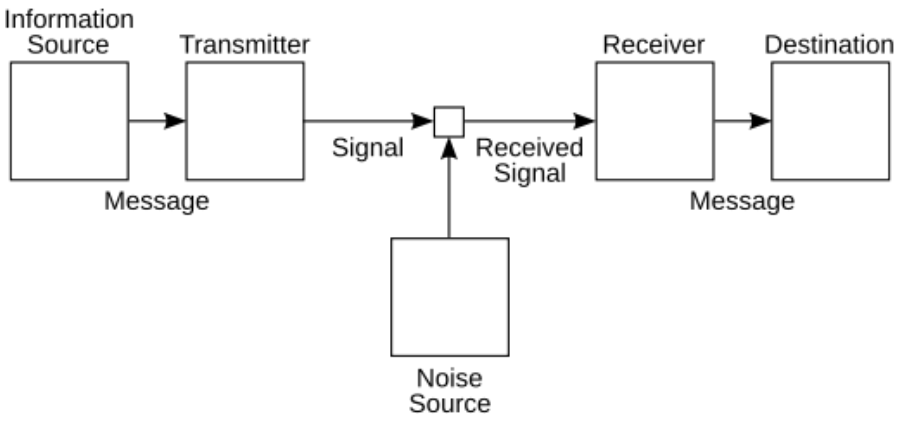


Figure 1.1: Classical Shannon-Weaver communication model [1], depicting the linear transmission process from source to destination through a noisy channel.

Decades later, Xin, Fan, and Letaief [2] revisited and expanded this perspective, bridging Shannon's conceptual intuition with the mathematical formalization of meaning in communication systems. Their work establishes the foundations of a *semantic information theory* and proposes a **three-layer communication framework** (*technical (syntactic)*, *semantic*, and *pragmatic*) that quantifies not only signal accuracy, but also meaning preservation and task effectiveness.

At the **technical level**, the focus lies on the accurate transmission of symbols through the channel, as originally formulated by Shannon. The **semantic level** extends this notion by emphasizing the correct interpretation of meaning, ensuring that the receiver reconstructs the same intent or message semantics as the sender. Finally, the **pragmatic level** considers the effectiveness and usefulness of the conveyed information for the specific task or decision-making process of the receiver.

This hierarchical view, illustrated in *Figure 1.2*, clarifies how semantic communication operates above the syntactic layer: successful transmission is no longer defined by perfect bit matching, but by the preservation of shared meaning and communicative intent between agents.

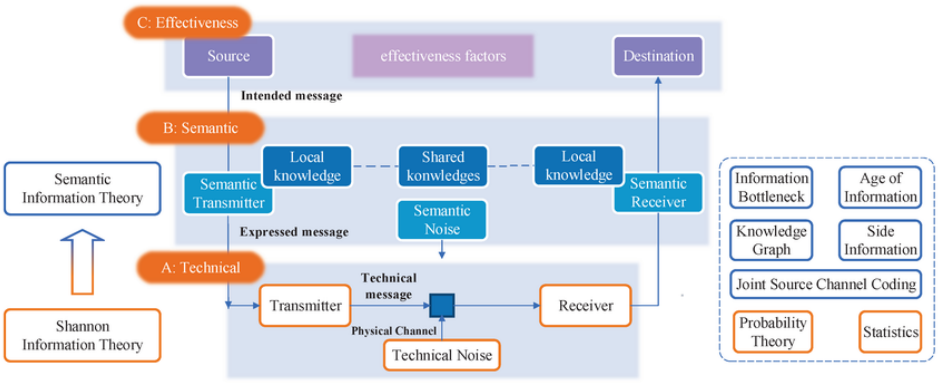


Figure 1.2: Three-layer communication framework proposed, illustrating the transition from syntactic to semantic and pragmatic information exchange. Source: [2]

In this context, the convergence of semantic theory and artificial intelligence has revitalized the study of communication, allowing machines to encode, interpret, and reconstruct meaning rather than mere symbols. This evolution paves the way toward intelligent, context-aware, and resource-efficient communication systems, which will be further explored in the following sections.

1.2 Research Objectives and Contributions

Recent research demonstrates that semantic communication can be effectively realized by leveraging advances in deep learning and natural language processing. In particular, Large Language Models (LLMs), built on transformer architectures, have shown remarkable abilities in understanding, compressing, and reconstructing natural language, capturing semantic dependencies that can be transmitted compactly while retaining meaning. In parallel, Knowledge Graphs (KGs) provide structured and interpretable representations of knowledge, grounding semantic content through a network built of explicit entities and relations. Their integration complements the contextual reasoning of LLMs with factual consistency, enabling a hybrid model for meaning-centered communication.

Building upon these directions, this thesis aims to design, implement, and evaluate a *Knowledge Graph-guided and LLM-based semantic communication system* capable of transmitting meaning efficiently over noisy or bandwidth-limited wireless channels. The proposed system operates through two transmission modes (token-based and embedding-based) allowing a comparative analysis between discrete and continuous representations under varying SNR conditions. Experimental evaluation is performed on the SST-2 dataset, including ablation studies to assess the impact of KG integration and decoder configurations.

The main scientific and technical contributions of this work can be summarized as follows:

- **End-to-end modular pipeline:** integrating semantic preprocessing, abstractive LLM encoding, physical-layer wireless transmission, and semantic decoding with refinement.
- **Hybrid knowledge extraction mechanism:** combining Stanford OpenIE and spaCy dependency parsing to ensure robust triplet generation even from unstructured text.
- **Dual-mode transmission framework:** supporting both token-based and embedding-based communication, enabling controlled trade-offs between compression and robustness.
- **Comprehensive benchmarking:** performed under realistic wireless conditions (Rayleigh fading, AWGN), evaluated through both physical metrics (compression ratio and bandwidth) and semantic metrics (BLEU, ROUGE, METEOR, BERTScore, Sentence similarity).

Through these objectives and contributions, the thesis demonstrates that transmitting meaning rather than bits leads to higher semantic fidelity and reduced transmission overhead, paving the way toward resilient and context-aware communication in constrained edge-network scenarios.

1.3 Thesis Outline

The remainder of this work is organized as follows:

- **Chapter 2 – Related Works:** presents a review of existing semantic communication research, from early neural approaches to LLM- and KG-based systems, identifying current limitations and motivation for this study.
- **Chapter 3 – Theoretical Background:** introduces the theoretical foundations of the thesis, covering semantic representations for text communication, knowledge graphs and relation extraction, transformer-based sequence-to-sequence models, and wireless channel models.
- **Chapter 4 – System Design and Architecture:** details the proposed framework, describing each stage of the pipeline from knowledge extraction and semantic encoding to decoding with contextual refinement.

- **Chapter 5 – Experimental Evaluation:** discusses datasets, configurations, and results across various SNR scenarios, highlighting the effects of KG integration, transmission mode (token-based vs. embedding- based), and different model configurations.
- **Chapter 6 – Conclusions:** summarizes the contributions and key takeaways of the thesis, discusses current limitations, and outlines directions for future research.

Chapter 2

Related works

Throughout the years, several directions have emerged in the development of semantic communication systems, each contributing different perspectives and solutions. The purpose of this chapter is to present a comprehensive overview of these research trajectories, analyzing their evolution, methodologies, and main findings. By reviewing these contributions, it becomes possible to highlight how each approach addresses the key challenges of modern wireless environments.

This analysis is organized into the following sections:

- Early deep learning-based semantic communication;
- LLM-enabled semantic transmission;
- Knowledge Graph Integration;
- Towards our proposed framework.

2.1 Early Deep Learning-Based Semantic Communication

Early research in semantic communication primarily explored the use of deep learning encoders and decoders capable of retaining the meaning relevant to the specific task. Han et al. [3] pioneered a semantic speech transmission model that optimized both communication and comprehension processes through a deep learning (DL) based transceiver, demonstrating strong resilience against channel noise. *Figure 2.1* illustrates their model.

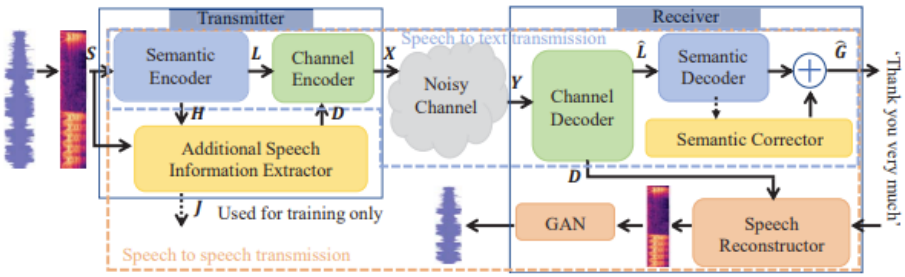


Figure 2.1: The system model for speech transmission. Source: [3]

Lokumarambage et al. [4] extended these concepts to the visual domain by designing an end-to-end semantic image transmission system, where convolutional encoders represented high-level semantic features rather than pixel-level details and a Generative Adversarial Networks (GAN) is used at the receiver as the transmission task to reconstruct the realistic image. The overall architecture is illustrated in *Figure 2.2*.

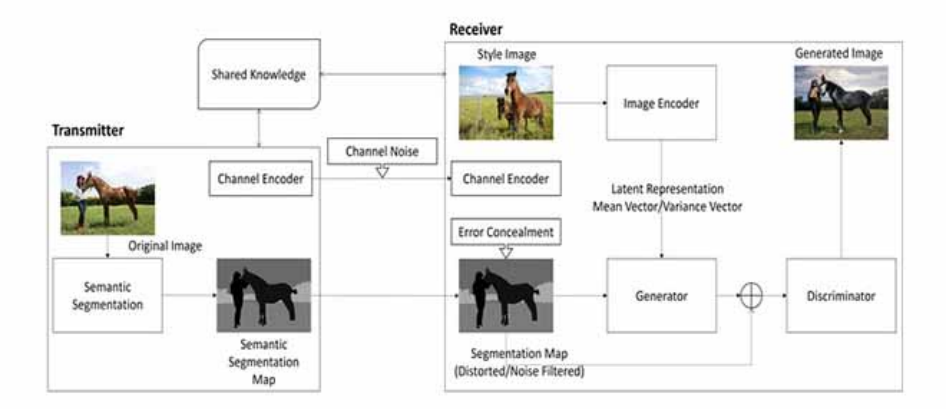


Figure 2.2: GAN high-level architecture for image transmission using semantic communication. Source: [4]

These early efforts collectively established the foundations of *semantic representation learning* as a cross-modal principle applicable to text, speech, and image data.

2.2 LLM-Enabled Semantic Transmission

With the advent of transformer architectures, *Large Language Models (LLMs)* have become central to semantic communication, especially for textual data. Their ability to capture contextual dependencies and generate meaning-aware representations makes them a natural fit for transmitting semantics rather than symbols over noisy wireless channels.

Wang et al. [5] first demonstrate the integration of large language models directly into the physical layer of semantic communication systems. In their work, the authors proposed an **LLM-enabled text transmission framework** in which both encoding and decoding are performed through prompt-based interactions with a pre-trained generative model. Like depicted in *Figure 2.3*, his approach effectively replaces conventional source and channel coding modules with a single process capable of producing compressed semantic representations of sentences. Their findings established the conceptual feasibility of leveraging LLMs as implicit encoders and decoders, opening the path to hybrid architectures that combine linguistic reasoning with structured semantic grounding.

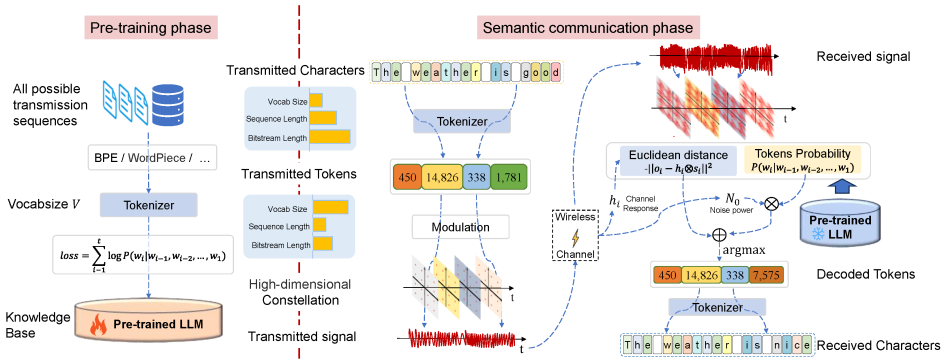


Figure 2.3: LLM-SC, an innovative LLM-enabled semantic communication system framework. Source: [5]

Building on this foundation, Chen et al. [6] further advanced the paradigm of LLM-driven semantic communication by introducing a hybrid architecture that combines **structured knowledge representation** and **retrieval-augmented generation** (RAG). Unlike earlier purely neural systems, their framework encodes messages as *semantic triples* (subject, predicate, and object) explicitly capturing relational meaning within transmitted content. These structured representations are then processed through an LLM-based generative model that reconstructs messages at the receiver side while leveraging external knowledge retrieval for contextual consistency. Moreover, by integrating retrieval mechanisms, the model supports personalization and adaptive decoding based on user context, marking a

step toward general-purpose semantic communication systems.

Although these works primarily emphasize accuracy and interpretability, subsequent studies have begun to address the computational and deployment challenges of LLMs in constrained environments. In this regard, Kalita [7] explored the application of large language models in resource-constrained edge environments, focusing on how semantic communication principles can enhance the efficiency of IoT networks. The study proposed an **edge-based semantic communication architecture** where lightweight LLMs are deployed at edge nodes to perform local semantic encoding and interpretation, thereby reducing the volume of data that must be transmitted to centralized servers. *Figure 2.4* presents the proposed edge-based semantic architecture, illustrating the interaction between local lightweight LLM agents and the cloud-level semantic controller.

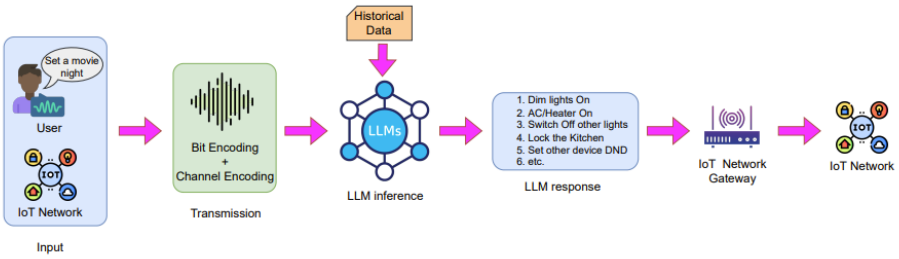


Figure 2.4: A framework enabling LLM-based semantic communication in Edge-based IoT system from [7].

Finally, Salehi et al. [8] proposed an **LLM-enabled end-to-end semantic communication pipeline** that fully integrates large language models into both the transmission and reconstruction stages. Their framework models the communication process as a continuous semantic transformation: the transmitter converts source text into a latent semantic representation through an instruction-based LLM prompt, which is then modulated and transmitted over a noisy channel. At the receiver side, another LLM reconstructs the message and performs semantic validation and refinement via post-decoding consistency checks, ensuring that the recovered output aligns with the original intent rather than its literal form. These studies collectively highlight the potential of LLMs to perform *semantic-aware compression and reconstruction.*

2.3 Knowledge Graph Integration

Parallel to LLM advancements, *Knowledge Graphs (KGs)* have emerged as a crucial component for improving context grounding and disambiguation. Jiang et al. [9] proposed one of the earliest frameworks that explicitly integrates structured knowledge into the semantic communication process. Their **KG-based semantic communication system** converts each sentence into a set of *subject-predicate-object* triplets, which are then transmitted according to their relative semantic importance, by assigning higher transmission priority and energy allocation to critical triplets under adverse channel conditions. Furthermore, the receiver reconstructs the message using KG reasoning, effectively compensating for missing or corrupted information. *Figure 2.5* illustrates this workflow, highlighting the semantic extraction module, traditional communication architecture, and semantic restoration module.

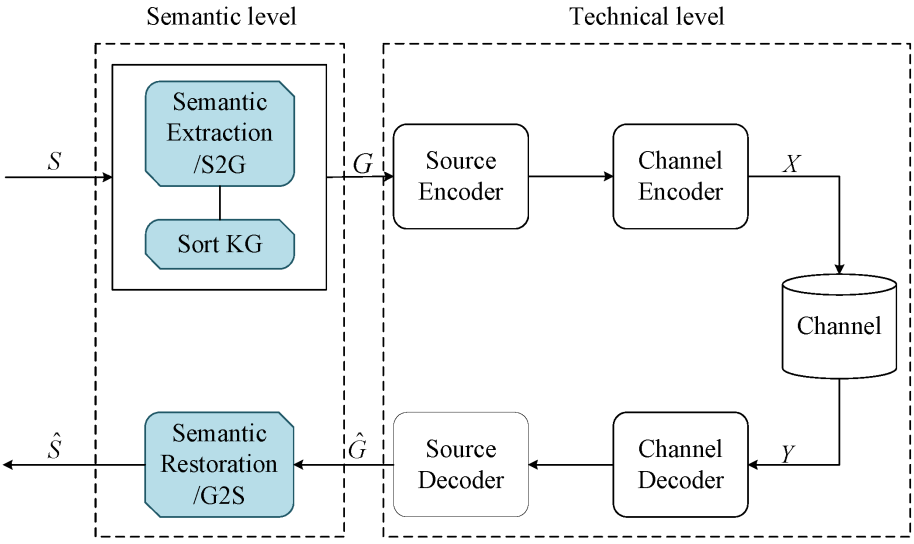


Figure 2.5: Overview of [9]: the structure of the proposed semantic communication system based on the knowledge graph.

Guo et al. [10] and Wang et al. [11] proposed complementary approaches toward integrating structured knowledge into semantic communication systems.

Specifically, Guo et al. developed a **large language model-driven knowledge graph construction scheme** designed for task-oriented semantic communication. Their framework leverages the generative and contextual reasoning capabilities of LLMs to automate the entire KG construction pipeline, from corpus collection and entity extraction to relation identification and incremental updates. By continuously refining entity–relation triples through dynamic learning, their approach ensures high recall and adaptability across communication scenarios, effectively

aligning transmitted semantics with domain-specific knowledge structures. *Figure 2.6* conceptually depicts this process, showing how an LLM-driven mechanism automates KG generation and updates through iterative learning.

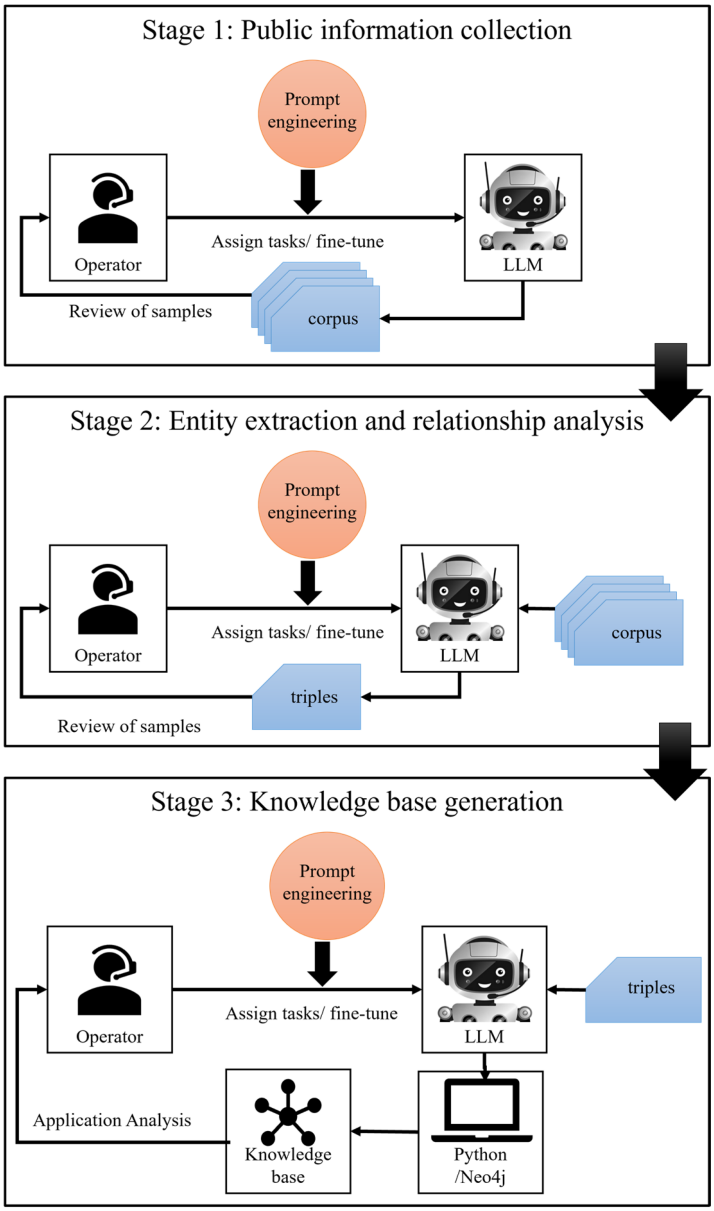


Figure 2.6: LLM-driven pipeline for automatic knowledge graph construction and semantic update. [10]

In contrast, Wang et al. introduced a **knowledge-enhanced semantic communication receiver** in which the decoding process is guided by factual triples retrieved from an external knowledge base. As visualized in *Figure 2.7*, their model performs semantic reasoning over these structured relations to infer missing or corrupted contextual elements during message reconstruction.

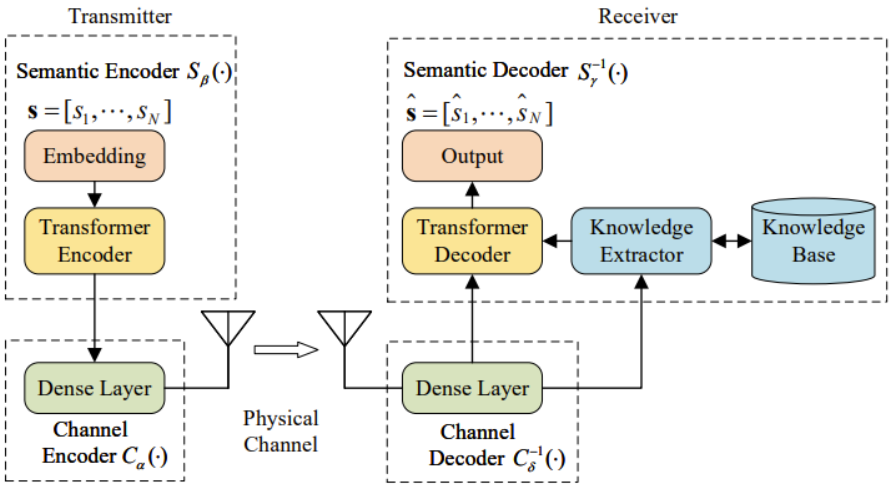


Figure 2.7: Knowledge-enhanced receiver proposed by [11].

Taken together, these studies demonstrate that **structured knowledge integration** can substantially enhance semantic consistency, contextual understanding, and robustness to distortion. This dual perspective directly motivates the hybrid approach proposed in the present work, where Knowledge Graph reasoning and LLM-based contextual encoding are jointly exploited within a unified end-to-end framework.

Later, Liang et al. [12] further advanced the integration of knowledge graphs into semantic communication by proposing the **KG-SemCom framework**, which explicitly aligns KG entities with message tokens and encodes information through a semantic fusion of contextual and knowledge-based representations. During decoding, KG-SemCom leverages relational logic and entity attributes to infer incomplete or distorted segments, effectively improving semantic recovery under low-SNR or noisy environments. *Figure 2.8* summarizes this fusion architecture, illustrating how contextual embeddings and KG entities are combined into a unified semantic representation.

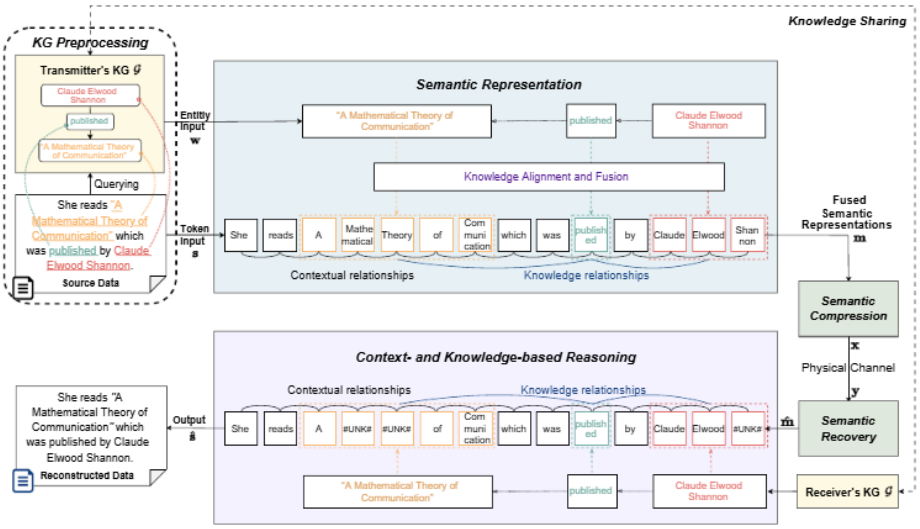


Figure 2.8: The framework of KG-SemCom. [12]

Building on these foundations, in addition to what said before, Salehi et al. [8] combined KG extraction with LLM-based encoding and decoding, achieving notable data compression while preserving semantic similarity.

2.4 Towards our proposed framework

The framework proposed in this study couples KG-based semantic grounding with LLM-driven encoding and decoding, establishing a coherent semantic layer resilient to channel defects such as fading and noise. Unlike earlier symbolic or token-only systems, it introduces a dual-mode architecture, token-based and embedding-based, that enables nice degradation across SNR regimes. Additionally, the model incorporates subword merging to mitigate semantic fragmentation and compares multiple transformer architectures (T5 and BART) within the same decoding pipeline.

A further innovation lies in the triplet extraction stage: the system first attempts knowledge extraction using Stanford OpenIE, providing accurate semantic triples directly from syntactic dependencies; if the OpenIE parser fails or yields incomplete results, a spaCy-based fallback module generates triples through predefined linguistic patterns, ensuring robust coverage across heterogeneous text inputs.

Collectively, these innovations address three central challenges identified in prior literature:

- **Semantic drift** during noisy transmission, mitigated through LLM-guided reconstruction and BERT-based refinement;

- **Robustness under Rayleigh fading**, tested via the dual token- and embedding-based transmission modes that ensure graceful degradation across SNR regimes;
- **Knowledge extraction reliability**, strengthened by the hybrid combination of OpenIE and spaCy triplet extraction pipeline, which ensures consistent graph quality even when confronted with unstructured or syntactically irregular input.

As a result, the proposed approach delivers a bandwidth-efficient and resilient semantic communication architecture, explicitly designed for constrained wireless environments where both meaning preservation and data compression are essential.

Chapter 3

Theoretical Background

While Chapter 1 motivates the shift from bit-level reliability to meaning preservation in modern wireless systems, and Chapter 2 reviews existing semantic communication architectures, this chapter introduces the theoretical foundations that underpin the framework proposed in this thesis.

The goal is to formalize the main semantic representations and models used in the subsequent system design and implementation. In particular, we focus on five core elements:

1. semantic representations for textual data;
2. knowledge graphs and relation extraction;
3. transformer-based sequence-to-sequence (seq2seq) models;
4. wireless channel models relevant to semantic communication;
5. semantic decoding and contextual refinement.

Throughout the chapter, we distinguish between *symbolic* communication, where success is defined by the accurate reconstruction of symbols, and *semantic* communication, where success is measured by the preservation of meaning, task performance, or both [2, 13].

3.1 Semantic Representations for Text Communication

The central object considered in this thesis is a short text sequence, such as a sentence from a sentiment analysis dataset. From the perspective of classical information theory, this sequence is treated as a string of discrete symbols to be

transmitted as faithfully as possible over a noisy channel. In semantic communication, the same sequence is viewed as a carrier of meaning, which may admit multiple equivalent textual realizations.

This section clarifies the distinction between symbolic and semantic transmission, introduces semantic representations for textual data, and discusses hybrid representations that combine symbolic, structured, and distributed views of meaning.

3.1.1 Symbolic vs. Semantic Transmission

In the classical Shannon model [1], a communication system is defined by an encoder f and a decoder g operating on a message random variable S. The encoder maps S into a channel input sequence $X = f(S)$, which is transmitted through a noisy channel producing an output Y. The decoder then outputs an estimate $\hat{S} = g(Y)$ of the original message. The design objective is to minimize the probability of symbol-level error:

$$\min_{f,g} \ \mathbb{P}\Big[\hat{S} \neq S\Big], \tag{3.1}$$

or equivalently to maximize the reliable transmission rate under constraints on bandwidth and power.

In this *technical* or *syntactic* view, two sentences are considered different as soon as they differ in at least one symbol, regardless of whether they convey the same meaning. For instance, the pair

"The movie was great." vs. "I really enjoyed the film."

would be regarded as a decoding error despite having nearly identical semantic content.

Semantic communication reframes the problem by explicitly introducing a random variable M representing the meaning, intent, or task-relevant information associated with the observed sentence S [2]. The mapping

$$\pi : S \mapsto M$$

may represent, for example, a latent semantic representation, a set of logical propositions, a point in an embedding space, or the label of a downstream task (such as sentiment). At the receiver, the goal becomes the accurate reconstruction of $\hat{M}$ rather than the exact recovery of S.

A generic *semantic distortion* measure can then be defined as

$$d_{\text{sem}}(M, \hat{M}) = 1 - \sigma_{\text{sem}}(M, \hat{M}), \tag{3.2}$$

where $\sigma_{\text{sem}}(M, \hat{M}) \in [0,1]$ denotes a semantic similarity score. In practice, σ_{sem} can be derived from:

- embedding-based similarity between sentences (e.g., cosine similarity between sentence embeddings);

Table 3.1: Symbolic vs. semantic transmission in text communication.

	Symbolic transmission	**Semantic transmission**
Primary object	Sequence of symbols (characters, tokens, bits)	Meaning, intent, or task-relevant information
Encoder output	Coded bitstream or modulated symbols	Compact semantic representation (e.g., triples, embeddings)
Success criterion	$\hat{S} = S$ (bit- or symbol-level accuracy)	$\hat{M} \approx M$ (semantic similarity, task performance)
Typical metric	BER, SER, mutual information	Semantic similarity, task accuracy, utility or reward
Example	Exact reconstruction of a sentence	Preserving sentiment or factual content despite paraphrasing

- symbolic overlap between sets of propositions;
- downstream task performance (e.g., probability of preserving sentiment labels).

The semantic design objective can thus be formulated as:

$$\min_{f,g} \ \mathbb{E}\Big[d_{\text{sem}}(M, \hat{M})\Big], \tag{3.3}$$

subject to constraints on bandwidth, power, or latency [14]. Importantly, the technical distortion $d_{\text{tech}}(S, \hat{S})$ and semantic distortion $d_{\text{sem}}(M, \hat{M})$ are related but not equivalent: low bit error rate does not guarantee preserved meaning, and conversely, two semantically equivalent sentences may differ substantially at symbol level.

Table 3.1 summarizes the conceptual difference between symbolic and semantic transmission, which will guide the design choices in the rest of the thesis.

In this thesis, we adopt an engineering-oriented definition of semantic communication: a system is deemed successful if it achieves low semantic distortion according to embedding-based similarity metrics and task-oriented performance, even in the presence of symbol-level errors or paraphrasing.

3.1.2 Semantic Representations for Textual Data

To operationalize the notion of meaning, it is necessary to choose a concrete form of semantic representation. For textual data, three complementary views are particularly relevant: symbolic, distributed, and structured representations.

Symbolic representations. At the most basic level, a sentence is modeled as a sequence of discrete tokens

$$s = (w_1, w_2, \ldots, w_n),$$

where w_i may denote words, subwords, or characters from a finite vocabulary. This representation is convenient for traditional source and channel coding, but it does not directly capture semantic similarity: small perturbations in the token sequence may correspond to drastic changes in meaning, and vice versa.

Distributed representations. Distributed representations embed words, or phrases, or entire sentences into a continuous vector space [15, 16]. Given a sentence s, a sentence encoder (e.g., a transformer-based model) produces an embedding

$$\mathbf{e}(s) \in \mathbb{R}^d,$$

such that geometrical proximity reflects semantic similarity. A common similarity measure is the cosine similarity:

$$\sigma_{\text{emb}}(s, \hat{s}) = \frac{\mathbf{e}(s)^\top \mathbf{e}(\hat{s})}{\|\mathbf{e}(s)\|_2 \, \|\mathbf{e}(\hat{s})\|_2}. \tag{3.4}$$

Distributed representations form the basis of many modern semantic metrics and are particularly well aligned with deep neural models, including large language models.

Structured representations. Structured representations explicitly capture entities, relations, and events present in the sentence. Examples include logical forms, Abstract Meaning Representations (AMR), and knowledge graphs [17]. In the context of semantic communication, a common choice is to represent textual content as a set of subject-predicate-object triples (s, p, o) and to aggregate these triples into a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$.

Each representation exposes different aspects of meaning: symbolic sequences are closest to the raw signal, distributed embeddings capture graded semantic similarity, and structured forms offer interpretability and logical consistency. A key design choice in semantic communication is how to combine these views into a representation that is both compact and robust to channel impairments.

3.1.3 Hybrid Semantic Representations

Recent research in semantic communication and natural language understanding has highlighted the benefits of hybrid representations that combine symbolic, structured, and distributed components [18, 12]. Conceptually, such representations exploit:

- *symbolic structure*, to retain the discrete nature of language and enable alignment with classical coding and transmission schemes;

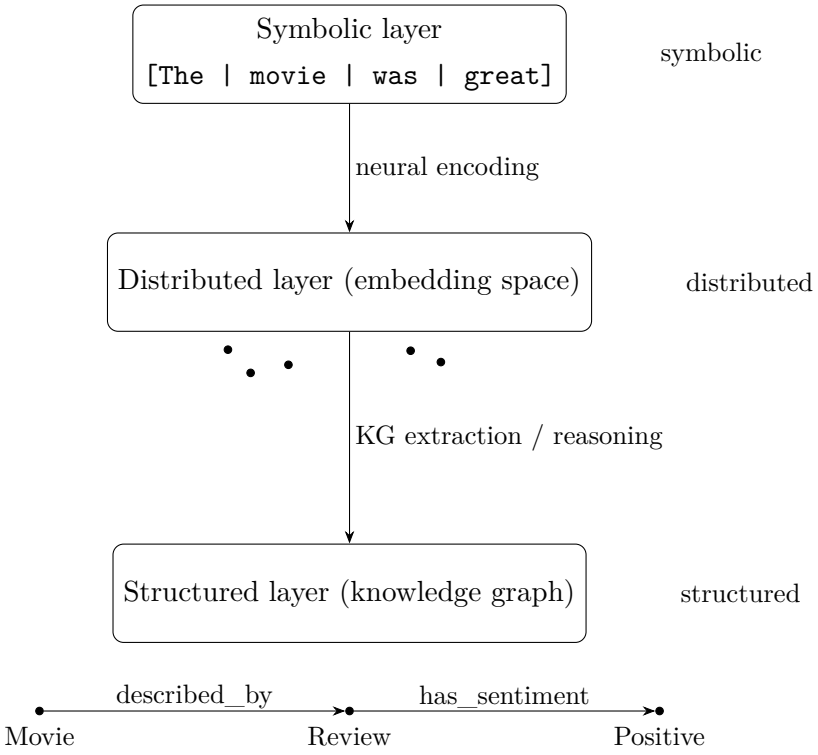


Figure 3.1: Conceptual illustration of hybrid semantic representations for text.

- *structured knowledge*, to represent explicit entities, relations, and constraints that support reasoning and disambiguation;
- *distributed embeddings*, to capture contextual nuances and paraphrastic variability in a continuous space.

A generic hybrid representation of a sentence s can be expressed as a tuple

$$R(s) = \Big(s,\ \mathcal{G}(s),\ \mathbf{e}(s)\Big),$$

where s is the symbolic sequence, $\mathcal{G}(s)$ is a graph (or set of triples) derived from s, and $\mathbf{e}(s)$ is a distributed embedding. Depending on system constraints, only a subset of these components may be explicitly transmitted, while the others can be reconstructed, approximated, or retrieved at the receiver.

Figure 3.1 conceptually illustrates these complementary views: the same sentence can be seen as a token sequence, as a point in an embedding space, and as a set of nodes and edges in a graph. Semantic communication systems can select, combine, or transform these layers to meet application-specific constraints on bandwidth, latency, and reliability.

In the remainder of this chapter and in Chapter 4, these general ideas will be instantiated using concrete tools for structured knowledge representation (knowledge graphs), neural encoding (transformer-based language models), and wireless transmission. Section 3.2 reviews knowledge graphs and relation extraction, Section 3.3 introduces transformer-based seq2seq encoding, Section 3.4 discusses the wireless channel models considered in this work, and Section 3.5 focuses on semantic decoding and contextual refinement.

3.2 Knowledge Graphs and Relation Extraction

Knowledge Graphs (KGs) provide a structured representation of knowledge in the form of entities and relations, typically organized as labeled graphs or sets of triples [17]. When combined with relation extraction techniques, KGs offer a natural way to convert unstructured text into machine-interpretable semantic structures. In the context of semantic communication, they can act as an intermediate layer that distills the core factual and relational content of a message before transmission.

3.2.1 Fundamentals of Knowledge Graphs

Informally, a knowledge graph is a labeled graph whose nodes represent entities (e.g., people, places, concepts) and whose edges represent semantic relations between these entities (e.g., *bornIn*, *locatedIn*, *hasSentiment*). Formally, a KG can be defined as a tuple

$$\mathcal{G} = (\mathcal{V}, \mathcal{R}, \mathcal{T}), \tag{3.5}$$

where $\mathcal{V}$ is a set of entities, $\mathcal{R}$ is a set of relation types, and $\mathcal{T}$ is a set of triples

$$\mathcal{T} \subseteq \mathcal{V} \times \mathcal{R} \times (\mathcal{V} \cup \mathcal{L}), \tag{3.6}$$

with $\mathcal{L}$ denoting a set of literal values (numbers, strings, dates, ...). Each triple $(s, p, o) \in \mathcal{T}$ can be interpreted as a directed, labeled edge from subject s to object o with relation label p.

KGs can be instantiated using different concrete data models, such as RDF (Resource Description Framework) graphs, property graphs, or hypergraphs, but the basic idea of representing knowledge as interconnected entities and relations remains the same [17]. In practice, KGs may also include:

- a *schema* or ontology that constrains which relations can connect which types of entities;
- metadata such as provenance, temporal qualifiers, or confidences associated with triples;

- logical axioms that allow inference of new triples from existing ones.

In semantic communication, KGs play two key roles. First, they provide an interpretable representation of the message content that can be transmitted in compressed form, for example by sending only the most important triples. Second, they enable reasoning and consistency checks at the receiver, helping to compensate for missing or corrupted information and to preserve the intended semantics even in the presence of noisy transmission.

3.2.2 Open Information Extraction (OpenIE)

To build KGs from raw text, it is necessary to extract relational tuples directly from natural-language sentences. Open Information Extraction (OpenIE) is a paradigm designed for this purpose [19]. Unlike traditional relation extraction, which targets a predefined set of relation types and requires annotated training data for each relation, OpenIE aims to discover arbitrary relations in a domain-independent way.

Given an input corpus $\mathcal{C}$ consisting of sentences $s \in \mathcal{C}$, an OpenIE system produces a set of relational tuples

$$\mathcal{T} = \bigcup_{s \in \mathcal{C}} \mathrm{OIE}(s), \tag{3.7}$$

where $\mathrm{OIE}(s)$ returns one or more triples (s_i, p_i, o_i) that capture relations expressed in the sentence. Early systems such as TextRunner and its successors [20, 21] relied on shallow syntactic features and lexical patterns, while later approaches incorporate dependency parsing, semantic role labeling, or neural models.

A typical OpenIE pipeline includes the following steps:

1. **Preprocessing**: tokenization, POS tagging, NER, and (optionally) dependency parsing;

2. **Clause or argument identification**: segmentation of the sentence into clauses and detection of candidate argument spans;

3. **Relation phrase detection**: identification of verbal or nominal predicates connecting arguments;

4. **Tuple construction and scoring**: assembly of triples (s, p, o) and assignment of confidence scores.

For example, consider the sentence:

"The movie received highly positive reviews from critics."

An OpenIE system might extract tuples such as

$$(\text{movie}, \text{received}, \text{highly positive reviews}), \quad (\text{reviews}, \text{from}, \text{critics}).$$

These tuples can subsequently be normalized or canonicalized (e.g., mapping *received* to *hasReview* and *highly positive* to a sentiment category). Modern OpenIE systems, such as Stanford OpenIE [22], leverage dependency parses to obtain more accurate argument boundaries and to handle complex constructions, including nested clauses and prepositional phrases.

In the context of this thesis, OpenIE provides a scalable mechanism for extracting SPO triples from arbitrary text, which are then used as building blocks for knowledge graph construction and semantic summarization.

3.2.3 Dependency Parsing and Pattern-Based Extraction

While OpenIE systems aim to be domain-independent and largely self-supervised, their performance in practice often relies on syntactic analysis and handcrafted patterns that exploit the structure of dependency trees [22, 23].

A dependency parse represents a sentence as a directed tree

$$D = (V, A, \ell), \tag{3.8}$$

where V is the set of tokens, $A \subseteq V \times V$ is a set of directed arcs, and $\ell : A \to \mathcal{L}_{\text{dep}}$ assigns a dependency label (e.g., `nsubj`, `dobj`, `amod`, `prep`) to each arc. Dependency parsing answers the question "who does what to whom", which is central for relation extraction.

Industrial-strength NLP libraries such as spaCy [24, 25] provide efficient implementations of POS tagging, NER, and dependency parsing, making it feasible to run relation extraction pipelines over large corpora. Using dependency trees, one can implement pattern-based extraction rules such as:

- verb-centered patterns that detect subject and object via `nsubj` and `dobj` dependencies;
- copular constructions that link a subject to a predicate nominal via `cop` and `attr`;
- prepositional relations that map structures like `prep` + `pobj` to binary relations.

For instance, the sentence

"The movie was directed by Christopher Nolan."

may yield a dependency structure where *movie* is the subject, *directed* is the main verb, and *Christopher Nolan* appears as the object of a prepositional phrase headed by *by*. A simple pattern that looks for `nsubj`-verb-`agent/by` relations can then extract the triple

$$(\text{movie}, \text{directed\_by}, \text{Christopher Nolan}).$$

In many practical systems, OpenIE and pattern-based extraction are combined in a hybrid approach: OpenIE provides broad coverage by proposing candidate tuples, while dependency-based rules are used to refine or supplement these tuples in cases where the OpenIE system fails, produces incomplete arguments, or mislabels relations. This hybrid strategy improves robustness on noisy or syntactically irregular text, which is particularly relevant for semantic communication over real-world datasets.

3.2.4 Triples as Units of Semantic Compression

From the perspective of semantic communication, SPO triples extracted from a sentence offer a compact and interpretable summary of its core meaning. Rather than transmitting the entire surface form of a sentence, a system may choose to transmit only a selected subset of triples, or a textual summary derived from them, thereby achieving semantic compression.

Let s denote an input sentence and let $\mathcal{T}(s)$ be the set of triples extracted from s through a combination of OpenIE and dependency-based patterns. A simple notion of *textual compression ratio* based on length can be defined as

$$\rho_{\text{text}}(s) = \frac{\text{length}(\text{summary}(s))}{\text{length}(s)}, \quad 0 < \rho_{\text{text}}(s) \leq 1, \tag{3.9}$$

where $\text{summary}(s)$ is a textual reconstruction derived from the triples (e.g., by linearizing the most salient SPO tuples). Alternatively, one may consider a *triple density* measure

$$\rho_{\text{triple}}(s) = \frac{|\mathcal{T}(s)|}{\text{length}(s)}, \tag{3.10}$$

which quantifies how many relational facts are extracted per token of input.

Not all triples contribute equally to the semantics of a sentence. In practice, triples can be ranked or filtered according to:

- extraction confidence scores produced by the OIE system;
- entity types (e.g., named entities vs. generic noun phrases).

Figure 3.2 illustrates a conceptual pipeline in which an input sentence is transformed into a set of SPO triples and then aggregated into a knowledge graph. In subsequent chapters, such triples will serve as semantic anchors that can be transmitted, stored, or used to summary the input sentence.

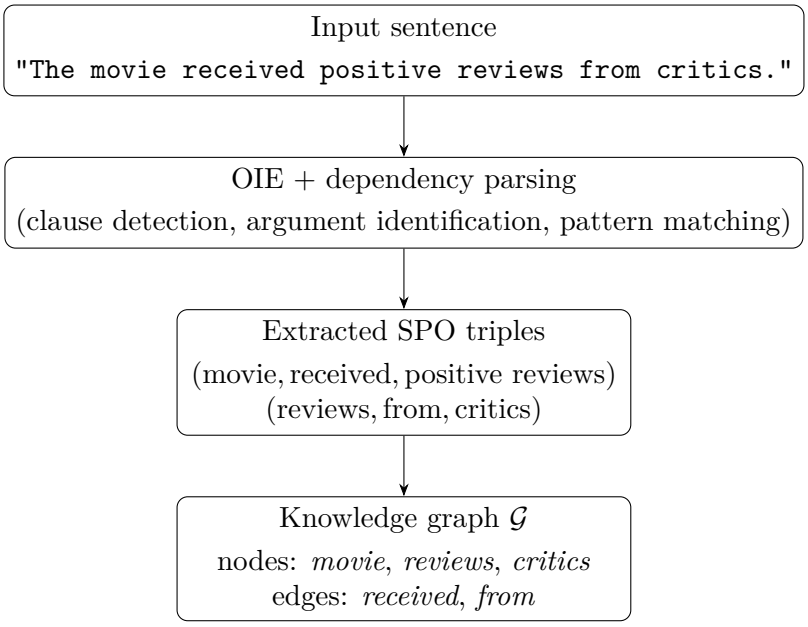


Figure 3.2: Conceptual pipeline for knowledge graph construction from text.

3.3 Transformer-Based Semantic Encoding

Transformers have become the dominant architecture for natural language processing, thanks to their ability to model long-range dependencies through self-attention and to support highly parallelizable training [26]. In semantic communication, transformer-based sequence-to-sequence (seq2seq) models provide a natural mechanism for encoding text into compact semantic representations and reconstructing it at the receiver.

3.3.1 The Transformer Architecture

The transformer architecture, introduced by Vaswani et al. in *Attention Is All You Need* [26], replaces recurrent and convolutional structures with multi-head self-attention and position-wise feed-forward networks. In its standard encoder-decoder formulation for text, a transformer consists of:

- an *encoder* that maps an input token sequence $(x_1, \ldots, x_n)$ into a sequence of contextual representations $\mathbf{H}_{\text{enc}} \in \mathbb{R}^{n \times d}$;
- a *decoder* that generates an output sequence $(y_1, \ldots, y_m)$ conditioned on $\mathbf{H}_{\text{enc}}$ and the previously generated tokens.

A high-level view of a transformer encoder–decoder architecture is shown in Figure 3.3.

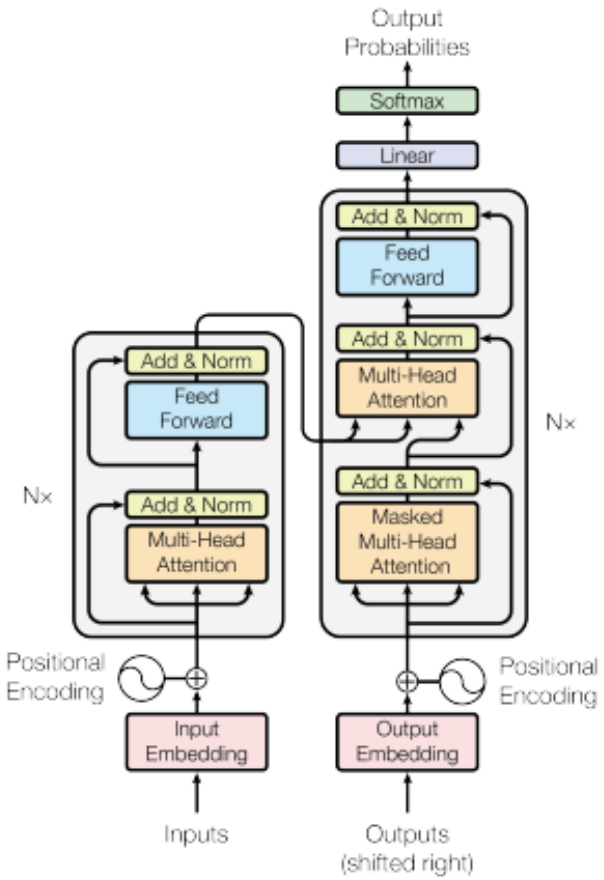


Figure 3.3: High-level encoder-decoder transformer architecture for sequence-to-sequence tasks. Source [26].

Each token x_i is first mapped to a vector embedding using a learned embedding matrix and combined with a positional encoding that injects information about token order. The result is a sequence of d-dimensional vectors that is passed through a stack of L identical encoder layers. Each encoder layer contains:

- a *multi-head self-attention* sub-layer, where each position in the sequence can attend to all other positions and aggregate information from them;
- a *position-wise feed-forward network* applied independently to each position.

Residual connections and layer normalization are applied around each sub-layer, which stabilizes training and allows the network to learn deep hierarchical

representations.

Intuitively, self-attention computes, for each token, a weighted mixture of all other tokens in the sequence, where the weights express how relevant one token is for another in the current context. Multi-head attention replicates this mechanism several times in parallel, so that different heads can specialize on different types of dependencies (e.g., syntactic relations, long-range semantic links, or local collocations).

The decoder mirrors the encoder architecture but includes two attention mechanisms. First, a *masked self-attention* layer allows each output position to attend only to previous positions, enforcing an autoregressive generation order. Second, an *encoder-decoder attention* layer lets the decoder attend over the encoder representations $\mathbf{H}_{\text{enc}}$, so that each generated token can leverage information from the entire input sentence.

Figure 3.4 provides a conceptual illustration of a transformer-based encoder-decoder model used for semantic encoding and decoding.

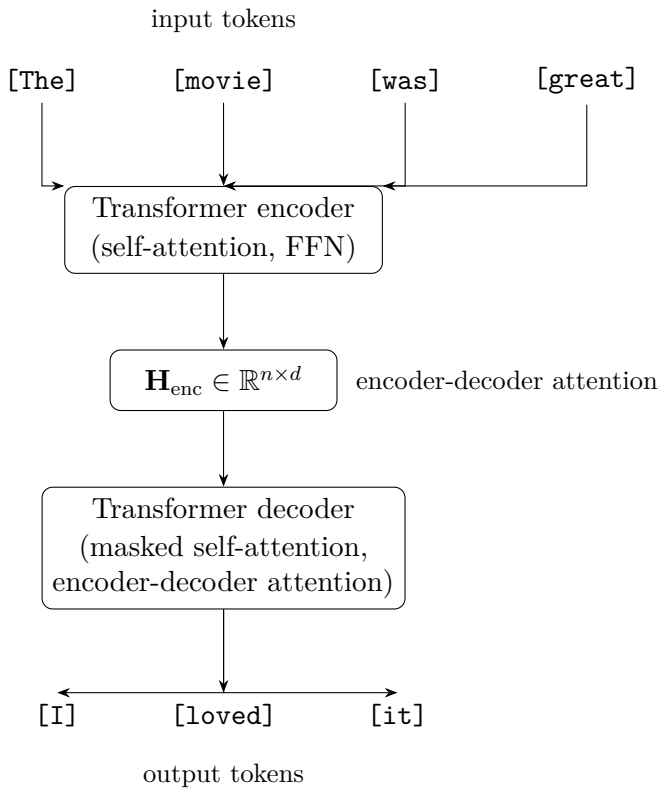


Figure 3.4: Conceptual view of a transformer-based encoder-decoder model for text.

3.3.2 Sequence-to-Sequence Encoding for Text

Sequence-to-sequence learning formalizes tasks in which an input sequence is mapped to an output sequence of (possibly) different length. Typical examples include machine translation, abstractive summarization, and style transfer. Originally implemented with recurrent neural networks [27], seq2seq models are now dominantly based on transformers due to their superior parallelization and performance [26].

Given an input token sequence $\mathbf{x} = (x_1, \ldots, x_n)$ and a target sequence $\mathbf{y} = (y_1, \ldots, y_m)$, a transformer-based seq2seq model first encodes $\mathbf{x}$ into $\mathbf{H}_{\text{enc}}$ and then autoregressively predicts each y_j by attending to both the previously generated tokens and the entire encoder representation. During training, the model is typically optimized with a cross-entropy loss over a large corpus of input-output pairs, using teacher forcing to stabilize learning.

In this work we focus on two widely used seq2seq transformers, T5 and BART [28, 29]. T5 is an encoder-decoder model trained in a "text-to-text" framework, where every task is cast as feeding text into the encoder and generating text from the decoder. BART combines a transformer bidirectional encoder, like BERT, and autoregressive decoder, like GPT, and is trained as a denoising autoencoder: the input sentence is corrupted by noise (e.g., token masking, deletion, or shuffling) and the model learns to reconstruct the original text. Figure 3.5 provides a schematic view of the T5 encoder-decoder architecture, highlighting the symmetry between encoder and decoder. On the other hand, figure 3.6 illustrates the BART architecture, which combines a bidirectional encoder with an autoregressive decoder and is pre-trained with a family of denoising objectives.

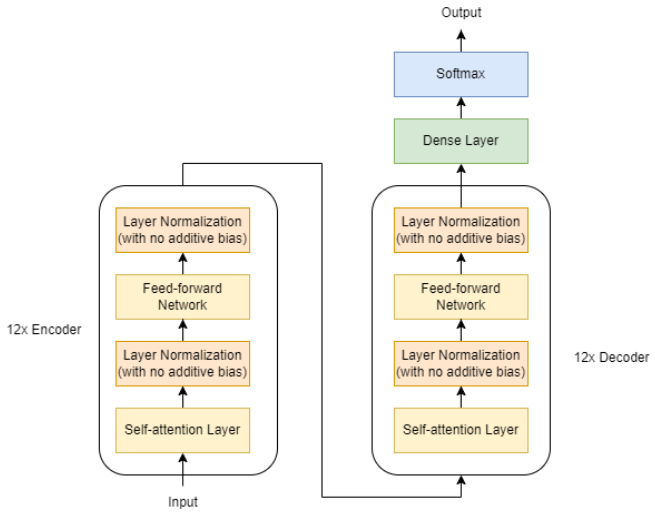


Figure 3.5: Schematic view of the T5 encoder-decoder architecture. Source [30].

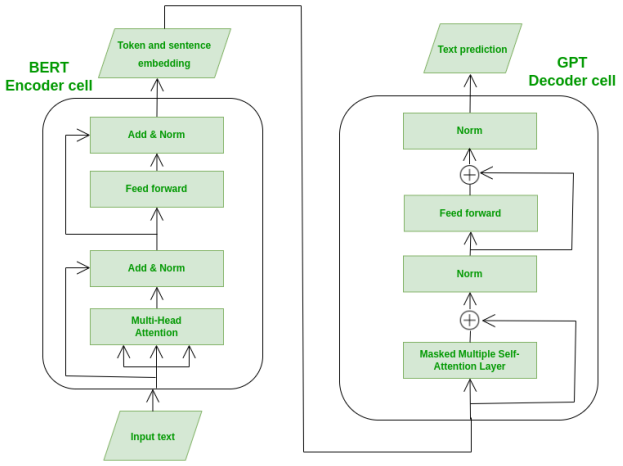


Figure 3.6: High-level architecture of BART. Source [31].

Both models are particularly suited to semantic communication because they:

- can generate paraphrases and summaries that preserve meaning while shortening or rephrasing the original text;
- are robust to noisy or partially corrupted inputs thanks to their training objectives.

In the proposed framework, we exploit these properties to implement a semantic encoder that transforms raw sentences into compact semantic representations and a semantic decoder that reconstructs meaning at the receiver.

3.3.3 Token-Based vs. Embedding-Based Semantic Representations

Transformers naturally provide two complementary levels of representation that are both relevant for semantic communication:

1. **Token-based representation:** the sequence of discrete output tokens $\mathbf{y} = (y_1, \ldots, y_m)$ produced by the decoder.
2. **Embedding-based representation:** the continuous hidden states $\mathbf{H}_{\text{dec}} \in \mathbb{R}^{m \times d}$ (or a subset thereof) associated with these tokens.

From a communication perspective, transmitting $\mathbf{y}$ corresponds to sending discrete indices from a vocabulary $\mathcal{V}$, which can be encoded into bitstreams using standard source and channel coding techniques. In contrast, transmitting $\mathbf{H}_{\text{dec}}$

Table 3.2: Comparison between token-based and embedding-based semantic representations.

	Token-based	**Embedding-based**
Representation	Discrete token IDs $y_j \in \mathcal{V}$	Continuous vectors $\mathbf{h}_j \in \mathbb{R}^d$
Encoding	Standard source/channel coding (bits, modulation)	Quantization or analog mapping of real-valued components
Granularity of errors	Symbol-level (token substitutions, insertions, deletions)	Perturbations in vector space (additive noise, distortion)
Interpretability	High (tokens map directly to text)	Lower; requires a decoder model to interpret
Robustness to small perturbations	Typically low: any bit flip can change a token	Often higher: small changes may not alter semantics
Storage / bandwidth	Efficient for short sequences; cost grows with m	Cost proportional to $m \times d$ (can be large)

corresponds to sending real-valued vectors, which may require quantization or analog modulation schemes, but can offer different robustness properties.

Table 3.2 summarizes the conceptual trade-offs between token-based and embedding based representations.

In many practical semantic communication systems, it is possible to transmit a compressed token sequence (e.g., an abstractive summary) or a compressed embedding representation (e.g., by dimensionality reduction or selective projection). The choice depends on the available bandwidth, the desired level of interpretability, and the robustness requirements of the application.

3.3.4 Semantic Compression and Robustness

Natural language is highly redundant: many tokens contribute little to the core semantics of a message. Transformer-based seq2seq models can exploit this redundancy by generating shorter but semantically equivalent sequences (e.g., summaries) or by encoding the meaning of a sentence into a fixed-size latent representation. This leads to the notion of *semantic compression*.

Let n be the length of the original token sequence and m the length of the compressed sequence produced by a semantic encoder. A simple token-level compression

ratio is

$$\rho_{\text{tok}} = \frac{m}{n}, \qquad 0 < \rho_{\text{tok}} \leq 1. \tag{3.11}$$

Similarly, when working with embeddings, one may consider a dimensionality reduction operator $\Phi : \mathbb{R}^{m \times d} \to \mathbb{R}^{m \times d'}$ with $d' < d$, and define an embedding-level compression ratio

$$\rho_{\text{emb}} = \frac{md'}{nd}. \tag{3.12}$$

The key requirement for semantic compression is that the semantic distortion $d_{\text{sem}}(M, \hat{M})$ remains small despite reductions in sequence length or dimensionality. Transformer-based models are well suited for this purpose: their contextualized representations capture global sentence-level information, and their attention mechanisms can focus on semantically salient tokens, allowing less informative parts of the input to be discarded or paraphrased.

Regarding robustness, semantic representations in embedding space exhibit a form of continuity: small perturbations to $\mathbf{H}_{\text{enc}}$ or $\mathbf{H}_{\text{dec}}$ (e.g., due to channel noise) may lead to outputs that are still semantically close to the original, especially when the decoder is trained to handle noisy or corrupted inputs. This contrasts with purely symbolic representations, where even a single bit error can drastically alter the decoded token sequence.

3.4 Wireless Channel Models for Semantic Communication

Semantic communication systems ultimately operate over physical wireless channels, which introduce attenuation, fading, and noise. To analyse the behaviour of the proposed framework under realistic yet tractable conditions, this thesis adopts a standard flat Rayleigh fading model with additive white Gaussian noise (AWGN). In this section, we summarize the corresponding complex baseband representation and discuss its implications for discrete (token-level) and continuous (embedding-level) semantic representations.

3.4.1 Intuitive View of Wireless Channels and Noise

Before introducing formal models, it is useful to build an intuitive picture of what happens to a signal travelling over a wireless channel.

Consider a simple baseband signal, for instance a sinusoid or a sequence of rectangular pulses. At the transmitter, this waveform is clean and perfectly known. During propagation, however, the signal encounters several effects:

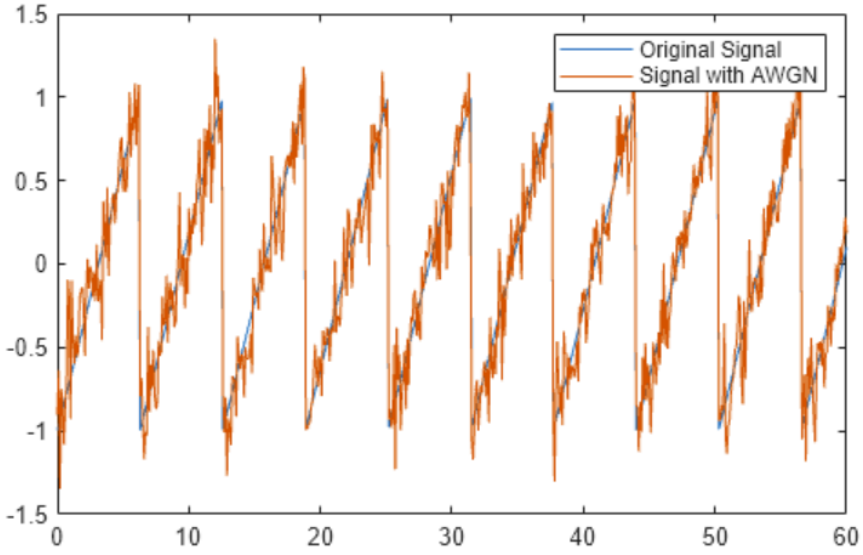


Figure 3.7: Example of a baseband signal (blue) corrupted by additive white Gaussian noise (orange). Source: [34].

- **Multipath and fading:** the signal reaches the receiver through multiple paths (reflections on buildings, vehicles, walls, etc.). The different copies may add constructively or destructively, causing the received amplitude to fluctuate in time. When destructive interference dominates, *deep fades* occur and the signal can almost disappear.

- **Noise:** every receiver front-end adds random fluctuations originating from thermal noise and other sources. This noise is often well modeled as *additive white Gaussian noise* (AWGN): it perturbs the waveform in an apparently erratic way.

Figure 3.7 illustrates this effect on a simple waveform: the original clean signal is shown together with a noisy version after passing through an AWGN channel at moderate SNR. Even though the overall shape is still recognizable, individual samples are visibly perturbed.

In the context of semantic communication, these physical-layer impairments translate into distortions of whatever representation is being transmitted: discrete symbols corresponding to tokens, or continuous-valued samples corresponding to embedding components. The next subsections formalize this behaviour using the standard flat Rayleigh fading plus AWGN model, which is widely adopted in wireless communication theory [32, 33].

3.4.2 Discrete and Continuous Encoded Signals

In a flat fading channel, the received complex baseband symbol y_k at discrete time index k can be written as

$$y_k = hx_k + n_k, \tag{3.13}$$

where x_k and y_k denote the transmitted and received symbols, $h \in \mathbb{C}$ is a complex fading coefficient capturing attenuation and phase rotation, and n_k is complex AWGN with zero mean and variance N_0.

The baseband model in (3.13) is agnostic to the origin of the symbols x_k: any sequence of complex numbers can be mapped to the physical channel. In a semantic communication framework based on transformers, two families of encoded signals are of particular interest:

1. **Discrete (token-level) signals.** The output of the semantic encoder is a sequence of discrete tokens, which can be mapped to a finite set of constellation points (e.g., BPSK, QPSK, or QAM symbols). In this case, x_k belongs to a discrete constellation, and channel impairments manifest themselves as symbol errors that may flip one token into another at the receiver.

2. **Continuous (embedding-level) signals.** The semantic encoder also produces continuous hidden representations, such as the rows of a hidden-state matrix $\mathbf{H} \in \mathbb{R}^{m \times d}$. These real-valued components can be mapped to channel symbols by treating them as samples of a continuous-time waveform or by serializing them into a stream of real or complex values. In this case, x_k is drawn from a continuous distribution, and channel impairments appear as additive perturbations in a high-dimensional vector space.

In both settings, the channel corrupts the semantic representation, but the nature of the corruption is different. For discrete constellations, errors are combinatorial (substitutions between constellation points), and the resulting performance is often summarized by bit error rate (BER) or symbol error rate. For continuous embeddings, errors are geometric perturbations, and their impact on meaning depends on the geometry of the embedding space and on the robustness of the decoder.

3.4.3 Impact on Token- and Embedding-Based Semantic Representations

The two types of encoded signals described above correspond directly to the token-based and embedding-based semantic representations introduced in Section 3.3.3. The flat Rayleigh fading model allows us to reason qualitatively about how fading and noise affect each of them.

Table 3.3: Impact of wireless channel impairments on discrete vs. continuous semantic representations.

	Discrete (token-level)	**Continuous (embedding-level)**
Physical mapping	Tokens → bits → constellation symbols	Embedding components → real or complex samples
Channel corruption	Bit and symbol errors; token substitutions and deletions	Additive perturbations in $\mathbb{R}^d$ or $\mathbb{C}^d$
Sensitivity to deep fades	High: a few errors may strongly change a token	Moderate: small perturbations may preserve semantics
Error control	Classical FEC and redundancy in token sequences	Robustness from embedding smoothness and decoder tolerance
Receiver processing	Equalization, hard decisions, bit-to-token mapping, LLM decoding	Equalization, reshaping, LLM decoding from noisy embeddings
Interpretability at receiver	Direct mapping from tokens to text	Requires a neural decoder to interpret embeddings

In the **token-based** case, each transmitted symbol encodes a finite number of bits that identify a token from a vocabulary. A single deep fade or a burst of noise can cause several bit errors, flipping the decoded token into a different one. From a semantic perspective, the impact of such errors can be very uneven: some token substitutions are benign (e.g., minor function words), while others drastically alter the meaning of the sentence (e.g., sentiment words or named entities). This motivates the use of error detection and correction mechanisms, as well as semantic refinement at the decoder to repair corrupted tokens.

In the **embedding-based** case, the transmitted object is a continuous vector representation that encodes the meaning of the sentence in a distributed fashion. Small perturbations of $\mathbf{H}$ due to fading and noise may still lead to similar outputs at the decoder, thanks to the continuity of the transformer mapping and the redundancy of the embedding space. Semantic distortion can then be related to continuous metrics such as mean squared error (MSE) or cosine similarity between original and corrupted embeddings, rather than to discrete bit errors.

Table 3.3 summarizes the main conceptual trade-offs between these two modes of transmission over a fading channel.

Figure 3.8 illustrates a simplified wireless communication chain with flat fading

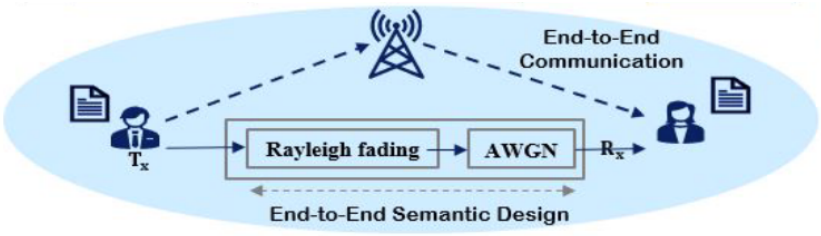


Figure 3.8: Simplified baseband chain for a noisy wireless channel. Source: [8].

and AWGN, highlighting the separation between the physical layer and the higher-layer semantic processing.

3.5 Semantic Decoding and Refinement

The final stage of the proposed semantic communication framework is the reconstruction of human-readable text at the receiver. In our design, semantic decoding is performed in two steps, both based on transformer models:

1. an *LLM-based semantic decoder* (T5 or BART) that maps corrupted tokens or embeddings produced by the wireless channel into an initial textual reconstruction;

2. a *refinement module* based on a masked language model (BERT) that detects and corrects local inconsistencies while preserving the global semantics [16].

This section describes how these two components work from a theoretical standpoint and how they relate to the notion of semantic distortion introduced in Section 3.1.1.

3.5.1 Semantic Decoder Based on T5 and BART

As discussed in Section 3.3, encoder-decoder transformers such as T5 and BART implement a conditional sequence-to-sequence mapping: given an input sequence or representation $\mathbf{u}$, the model generates an output sentence $\mathbf{y}$ that is consistent with it. This behaviour can be summarized as a conditional distribution

$$p_{\theta_{\text{dec}}}(\mathbf{y} \mid \mathbf{u}),$$

which is implemented through an encoder that processes $\mathbf{u}$ into contextual representations and a decoder that autoregressively predicts the output tokens.

In the proposed framework, the input $\mathbf{u}$ is not the clean source sentence, but a *noisy* representation $\tilde{\mathbf{z}}$ produced by the wireless channel and by the semantic encoder:

- in the **token-based mode**, $\tilde{\mathbf{z}}$ is a sequence of tokens corrupted by bit and symbol errors (Section 3.4.2);
- in the **embedding-based mode**, $\tilde{\mathbf{z}}$ is a matrix of continuous embeddings that has been perturbed by Rayleigh fading and AWGN in the baseband channel.

Token-based decoding. In token mode, the channel output is a sequence $\tilde{\mathbf{z}} = (\tilde{z}_1, \ldots, \tilde{z}_m)$ where some positions may be unreliable (e.g., flagged by a corruption mask derived from bit- or token-level checks). To make this information usable by the semantic decoder, uncertain positions are replaced by special mask symbols, while reliable tokens are kept as *anchors*. Let $\tilde{\mathbf{y}}$ denote the resulting sequence of anchors and masks.

A model such as BART or T5, pre-trained with denoising objectives [29, 28], is then used as a conditional language model that reconstructs a clean sentence from $\tilde{\mathbf{y}}$:

$$\hat{\mathbf{y}}^{(0)} \sim p_{\theta_{\text{dec}}}\Big(\mathbf{y} \mid \tilde{\mathbf{y}}\Big). \tag{3.14}$$

During generation, the decoder attends both to the unmasked tokens (which act as semantic and syntactic anchors) and to the internal encoder states, and fills in the masked positions so as to produce a fluent and coherent sentence. This matches closely the way T5 and BART are originally trained: they receive corrupted inputs (with spans deleted, permuted, or masked) and learn to reconstruct the original text.

Embedding-based decoding. In embedding mode, the channel output is a sequence of vectors $\tilde{\mathbf{z}} \in \mathbb{R}^{m \times d}$ obtained by transmitting hidden states (e.g., $\mathbf{H}_{\text{enc}}$) through the Rayleigh+AWGN channel described in Section 3.4. From a transformer viewpoint, these embeddings play the role of encoder representations: they summarize the meaning of the input sentence in a distributed way and are consumed by the decoder through cross-attention.

Conceptually, the semantic decoder now operates directly on the continuous representation:

$$\hat{\mathbf{y}}^{(0)} \sim p_{\theta_{\text{dec}}}\Big(\mathbf{y} \mid \tilde{\mathbf{z}}\Big), \tag{3.15}$$

where $\tilde{\mathbf{z}}$ replaces the clean encoder states that would be available in a standard seq2seq setting. The effect of the wireless channel is thus analogous to injecting noise into a latent layer of the model: if the embeddings are only mildly perturbed, the decoder can still recover a sentence with similar semantics; if the perturbation is too strong, the output drifts away from the intended meaning.

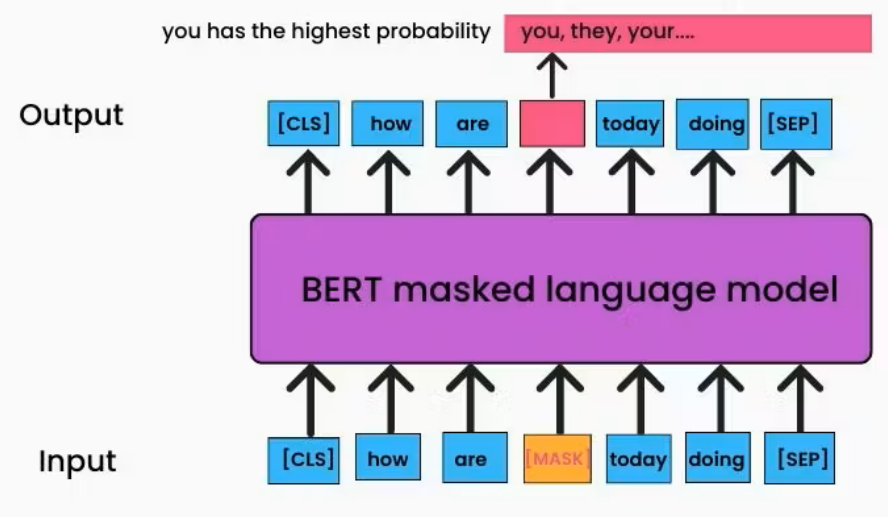


Figure 3.9: Illustration of BERT in a masked language modelling setup. Source [35].

In both token and embedding mode, the semantic decoder produces an initial reconstruction $\hat{\mathbf{y}}^{(0)}$ that aims to be close in meaning to the original sentence, even if the input representation has been corrupted by the physical channel.

3.5.2 Masked Language Modelling with BERT

The initial reconstruction $\hat{\mathbf{y}}^{(0)}$ produced by the semantic decoder may contain residual errors: wrong words, missing or repeated tokens, and local inconsistencies that are hard to fix solely from the noisy representation $\tilde{\mathbf{z}}$. To further improve the quality of the received text, the framework adopts a second-stage refinement based on BERT [16], a transformer encoder pre-trained with a masked language modelling (MLM) objective. A graphical overview of BERT used in a masked language modelling setup is shown in Figure 3.9, where one token is replaced by `[MASK]` and the model is trained to recover the original tokens from their bidirectional context.

In MLM, the model receives a sentence in which a subset of tokens has been replaced by a special `[MASK]` symbol, and it is trained to predict the masked tokens from their left and right context. Formally, let $\mathbf{w} = (w_1, \ldots, w_T)$ be a tokenized sentence and $\mathcal{M} \subseteq \{1, \ldots, T\}$ the set of masked positions. BERT is trained to maximize

$$\mathcal{L}_{\mathrm{MLM}}(\theta_{\mathrm{mlm}}) = \sum_{i \in \mathcal{M}} \log p_{\theta_{\mathrm{mlm}}}\Big(w_i \mid \mathbf{w}_{\setminus \mathcal{M}}\Big), \tag{3.16}$$

where $\mathbf{w}_{\setminus \mathcal{M}}$ denotes the unmasked tokens. This objective forces the model to learn rich bidirectional dependencies, making it well suited for local error correction.

Applied to the output $\hat{\mathbf{y}}^{(0)}$ of the semantic decoder, BERT is used in two conceptual steps:

1. **Suspicious token detection.** Identify positions likely to be erroneous or unreliable. In practice, these can be derived from:

   - the channel corruption mask (positions where tokens are known to have been corrupted at the physical layer);
   - simple heuristics over $\hat{\mathbf{y}}^{(0)}$ (e.g., repeated words, unlikely subwords, low-confidence tokens).

   The resulting set of indices defines the mask set $\mathcal{M}$.

2. **Masked prediction.** Replace tokens at positions $i \in \mathcal{M}$ with `[MASK]`, feed the sequence into BERT, and read off, for each masked position, a distribution over candidate replacements. The refined sentence $\hat{\mathbf{y}}^{(1)}$ is obtained by selecting the most plausible candidate at each masked position, while leaving the other tokens unchanged.

Figure 3.10 summarizes the overall two-stage decoding pipeline.

The next chapter instantiates this decoding scheme with specific models (T5, BART, BERT) and evaluates its behaviour under the Rayleigh fading and AWGN conditions described in Section 3.4, using semantic metrics to quantify the effect of refinement.

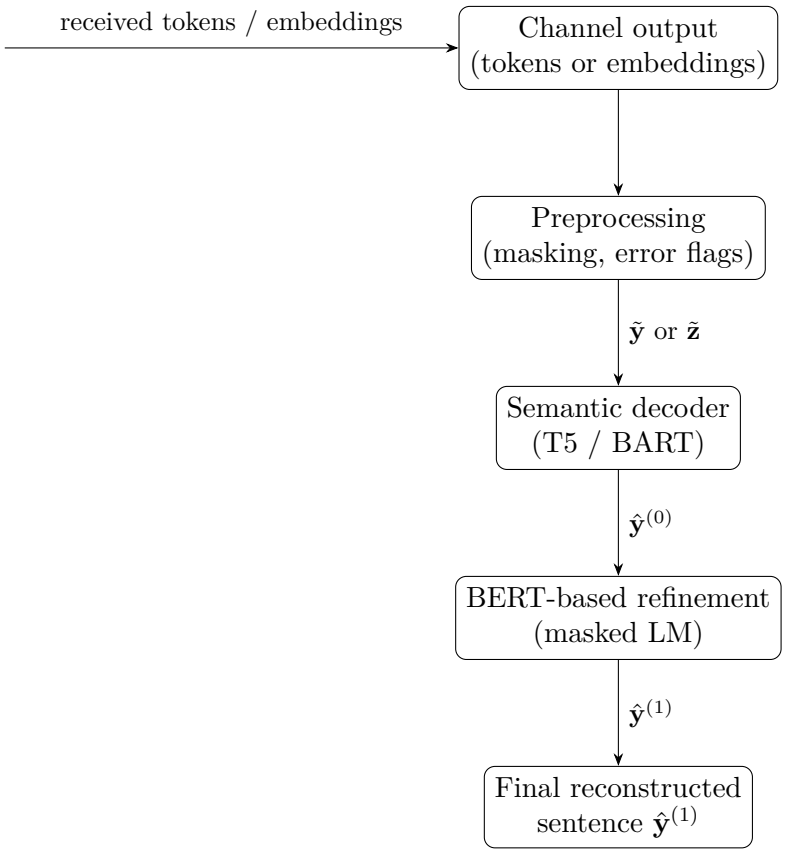


Figure 3.10: Two-stage semantic decoding pipeline.

Chapter 4

System Design and Architecture

4.1 Overview of the Proposed Framework

This chapter presents the concrete system design and implementation of the proposed KG-LLM semantic communication framework. While Chapter 3 has introduced the underlying concepts of semantic representations, knowledge graphs, transformer-based sequence-to-sequence models, and wireless channel models, the focus here is on how these components are instantiated and combined in an end-to-end pipeline.

At a high level, the system transforms an input natural-language sentence into a compact semantic representation that is transmitted over a noisy wireless channel and then reconstructed at the receiver with minimal loss of meaning. The architecture is organised into three main processing phases:

1. **Semantic preprocessing and knowledge extraction** (Phase 1), which analyses the input sentence, extracts structured semantic triples, and builds a sentence-level knowledge graph;

2. **LLM-based semantic encoding and channel mapping** (Phase 2), which produces a compressed semantic representation using a transformer encoder-decoder model and maps it either to discrete tokens or continuous embeddings for transmission over the wireless channel;

3. **Contextual decoding and semantic refinement** (Phase 3), which reconstructs the sentence at the receiver using an LLM-based decoder and a BERT-based refinement stage for semantic consistency.

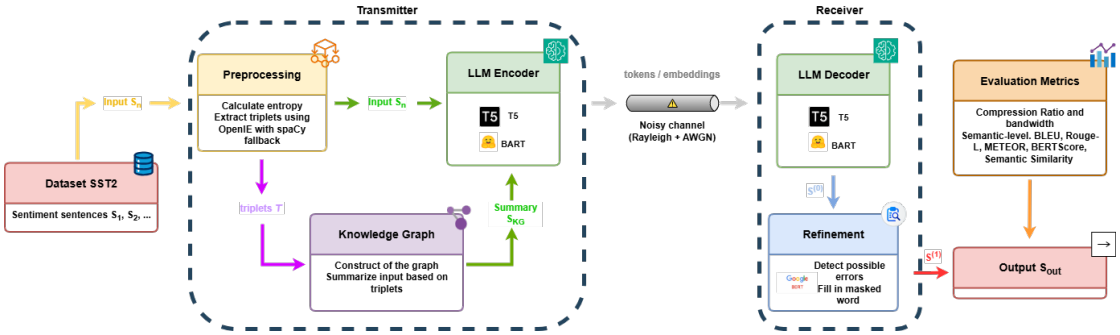


Figure 4.1: High-level block diagram of the proposed KG-LLM semantic communication pipeline.

These three phases are implemented as modular software components that can be configured, enabled, or bypassed independently. This modularity allows us to investigate different design choices (e.g., with or without KG guidance, token-based versus embedding-based transmission, or different LLM backends) while keeping the overall pipeline unchanged. The individual modules expose well-defined input/output interfaces so that they can be composed in a uniform end-to-end workflow.

A high-level block diagram of the proposed architecture is shown in Fig. 4.1. The diagram highlights the data flow between the semantic preprocessing block, eith the role of the knowledge graph, the LLM-based encoder, the wireless channel, and the semantic decoder.

4.1.1 Mapping Between Theory and Implementation

The design of the framework closely follows the theoretical components introduced in Chapter 3. The semantic preprocessing and KG construction block (Phase 1) operationalises the notions of entities, relations, and graph-based semantic representations discussed in Section 3.2. In practice, this block combines entropy-based sentence analysis with a hybrid extraction pipeline that leverages both OpenIE and a spaCy-based NLP stack to populate a sentence-level knowledge graph.

The LLM-based encoding block (Phase 2) instantiates the transformer encoder-decoder models described in Section 3.3. Depending on the configuration, the encoder can operate either in a predominantly text-driven mode (using only the original sentence) or in a KG-aware mode, where additional cues derived from the knowledge graph are included in the input. The resulting compressed representation is then mapped to either a sequence of discrete tokens or a sequence of continuous embeddings, directly corresponding to the symbolic and embedding-based semantic representations introduced in Section 3.1.

The wireless channel simulation block corresponds to the fading and noise models studied in Section 3.4. It injects Rayleigh fading, additive white Gaussian noise, and, when configured, multipath effects into the transmitted semantic representation. Finally, the semantic decoding and refinement block (Phase 3) realises the contextual decoding strategies of Section 3.5, combining LLM-based reconstruction and BERT-based masked language modelling into a single receiver-side pipeline.

4.1.2 Software Stack and Code Organization

The framework is implemented in Python using a set of widely adopted libraries for deep learning and natural language processing. Transformer-based models are provided by the Hugging Face ecosystem, which is used both for encoder-decoder LLMs (such as T5 or BART) and for the BERT-based refinement module. The semantic preprocessing pipeline relies on spaCy for tokenisation, part-of-speech tagging, dependency parsing, and named entity recognition, while OpenIE is used, when available, to extract additional candidate triples from each sentence.

Knowledge graphs are internally represented using graph data structures (e.g., NetworkX) and can be exported to RDF or OWL formats through dedicated libraries for serialisation when needed. The wireless channel is implemented as a standalone module that supports different fading types, SNR configurations, and transmission modes (token-based or embedding-based), but exposes a unified interface to the rest of the pipeline.

From a code-organization perspective, each major block of the architecture is encapsulated in a separate module. The semantic preprocessing and KG construction logic is grouped into the Phase 1 module; the LLM-based encoding and semantic compression procedures are implemented in the Phase 2 module; the channel model is encapsulated in a `wireless_channel` module; and the receiver-side semantic decoding and refinement constitute the Phase 3 module. This structure mirrors the block diagram in Fig. 4.1 and reflects the three-phase organisation outlined above, facilitating both reuse and experimental ablations.

4.1.3 Design Assumptions and Constraints

The system design is guided by a set of assumptions that reflect the targeted application scenarios and the practical limitations of the underlying models. First, the framework focuses on short to medium-length sentences, as typically found in sentiment analysis or intent classification datasets, rather than on long documents. This choice is aligned with the computational cost of transformer architectures and with the need to maintain low latency in wireless communication settings.

Second, the current implementation assumes English text and uses pretrained

models that have been primarily trained on English corpora. Extending the framework to multilingual scenarios would require appropriate multilingual LLMs and potentially language-specific adaptations of the semantic preprocessing pipeline.

Third, the wireless channel module is designed to emulate bandwidth-limited and noisy conditions using Rayleigh fading and AWGN models, with configurable signal-to-noise ratios (SNRs) and transmission modes. Rather than modelling a specific physical deployment, the channel is intended as a flexible abstraction that allows us to stress-test the semantic pipeline under different levels of degradation.

Finally, the framework is built around pretrained LLMs and does not rely on task-specific fine-tuning of large models, in order to keep the computational requirements compatible with edge or resource-constrained deployments. Where necessary, smaller variants of the base models can be selected, and the compression strength can be adjusted through decoding parameters such as maximum length or beam size. These assumptions and constraints will be revisited in the evaluation and discussion chapters, where their impact on performance and generality is analysed.

4.1.4 End-to-End KG-LLM Semantic Communication Algorithm

To summarise the interactions between the different modules introduced above, Algorithm 1 presents a high-level description of the end-to-end KG-LLM semantic communication pipeline. The algorithm explicitly distinguishes between the three main phases of the system: semantic analysis and knowledge graph construction at the transmitter, LLM-based semantic compression and channel mapping, and receiver-side reconstruction with contextual refinement.

This algorithmic view complements the block diagram in Fig. 4.1: while the figure emphasises the structural decomposition of the system into functional blocks, Algorithm 1 makes explicit the order of operations, the decision points (such as the choice between token-based and embedding-based transmission), and the flow of information between LLMs and knowledge graphs. The following sections provide a detailed description of each phase of the pipeline, linking the conceptual steps in Algorithm 1 to their concrete software implementation.

4.2 Phase 1 – Semantic Preprocessing and Knowledge Extraction

Phase 1 implements the first stage of the KG-LLM semantic communication pipeline, corresponding to the semantic analysis and knowledge graph construction block in Fig. 4.1 and to the first phase of Algorithm 1. Its goal is to transform each raw

Algorithm 1 KG-LLM End-to-End Semantic Communication

Require: Sentence S
Ensure: Reconstructed sentence S_{out}
Phase 1: Semantic Analysis and KG-based Summarisation
Compute entropy $E(S)$
if $E(S) \leq H_\theta$ **then** ▷ low-entropy: run full KG pipeline
Try OpenIE on S to obtain triples $\mathcal{T}_{\text{OIE}}$
if OpenIE is not available or $\mathcal{T}_{\text{OIE}}$ is empty **then**
Run spaCy-based pipeline on S to obtain triples $\mathcal{T}_{\text{spaCy}}$
$\mathcal{T} \leftarrow \mathcal{T}_{\text{spaCy}}$
else
$\mathcal{T} \leftarrow \mathcal{T}_{\text{OIE}}$
end if
Build a knowledge graph $G(S)$ from $\mathcal{T}$
Derive a KG-based condensed summary S_{KG} from $G(S)$
Set semantic input $S_{\text{sem}} \leftarrow S_{\text{KG}}$
else ▷ high-entropy: skip KG and summarisation
Set semantic input $S_{\text{sem}} \leftarrow S$
end if

Phase 2: LLM-Based Semantic Compression and Channel Mapping
Tokenize S_{sem} with the seq2seq LLM tokenizer
Obtain compressed representation $R \leftarrow \text{LLM\_encode}(S_{\text{sem}})$
if `transmission_mode` = "token" **then**
Derive compact token sequence Z from R
Map Z to a channel payload and transmit it
else if `transmission_mode` = "embedding" **then**
Derive embedding sequence H from R
Map H to a channel payload and transmit it
end if

Phase 3: Receiver-Side Reconstruction and Semantic Refinement
Receive noisy payload at the receiver
if `transmission_mode` = "token" **then**
Reconstruct token sequence $\hat{Z}$ and set $S^{(0)} \leftarrow \text{LLM\_decode}(\hat{Z})$
else if `transmission_mode` = "embedding" **then**
Reconstruct embedding sequence $\hat{H}$ and set $S^{(0)} \leftarrow \text{LLM\_decode}(\hat{H})$
end if
Refine $S^{(0)}$ with a BERT-based masked LM to obtain $S^{(1)}$
Set $S_{\text{out}} \leftarrow S^{(1)}$ and **return** S_{out}

input sentence into an enriched representation that combines lexical information, shallow statistics, and a structured view of the underlying entities and relations. In addition to building a sentence-level knowledge graph, Phase 1 also derives a compact, KG-driven textual summary of the input, which will be used as a simplified semantic input for the encoder in Phase 2.

Given an input sentence S, Phase 1 first computes an entropy-based score that characterises the lexical variability of the sentence and uses it to distinguish more structured, information-dense sentences from simpler ones. It then applies a hybrid relation extraction pipeline, combining Open Information Extraction (OpenIE) and a spaCy-based dependency parser, to obtain candidate triples of the form *(subject, relation, object)*. These triples are merged and filtered and are finally used to build a sentence-level knowledge graph $G(S)$.

4.2.1 Module Overview and I/O Contract

From an implementation perspective, Phase 1 is encapsulated in a dedicated module that exposes a simple input/output contract. The module receives a single natural-language sentence S as input and returns a structured object `ProcessedSentence`, which aggregates all the information needed by the subsequent phases.

At a high level, `ProcessedSentence` contains:

- the original sentence S and its tokenised form;
- the entropy score $E(S)$ and a Boolean flag indicating whether the sentence is classified as high- or low-entropy;
- the set of entities and candidate triples extracted by the hybrid OpenIE + spaCy pipeline;
- a sentence-level knowledge graph $G(S)$, represented as a directed labelled graph;
- a KG-driven textual summary S_{KG}, obtained by verbalising the consolidated triples into a shorter, semantically focused sentence;
- optional summary statistics (e.g., number of nodes and edges, average degree, number of triples retained).

This enriched representation is passed to Phase 2. When the sentence is processed through the KG branch, the encoder does not receive the original sentence S alone, but a simplified semantic input derived from S_{KG} (possibly combined with S), so that the LLM can focus on the most relevant entities and relations while operating on a shorter and more structured text.

4.2.2 Entropy-Based Sentence Analysis

The first operation performed in Phase 1 is an entropy-based analysis of the input sentence. Given a sentence S with tokens $\{t_i\}$, we compute a lexical entropy score $E(S)$ based on the empirical token distribution:

$$E(S) = -\sum_i p(t_i) \log_2 p(t_i), \tag{4.1}$$

where $p(t_i)$ denotes the relative frequency of token t_i within the sentence (or, when available, within a reference corpus). This definition corresponds to the classical Shannon entropy used in information theory [1].

The resulting entropy value is compared against a configurable threshold H_θ. Sentences with $E(S) > H_\theta$ are labelled as *high-entropy*, whereas sentences with $E(S) \leq H_\theta$ are labelled as *low-entropy*. Intuitively, low-entropy sentences tend to be shorter or more repetitive and are often easier to map to a small number of clear semantic relations (for instance, simple statements or polarity-bearing sentences), whereas high-entropy sentences may contain multiple clauses, modifiers, or contrasting opinions.

This classification does not change the overall structure of the pipeline, but controls whether the sentence is routed through the KG-based branch or treated as purely text-based. Sentences classified as low-entropy are passed to the KG extraction and summarisation steps described in Sections 4.2.3 and 4.2.4, whereas high-entropy sentences bypass these operations and are forwarded to Phase 2 with the original text as semantic input.

4.2.3 Hybrid Triple Extraction with OpenIE and spaCy

After the entropy-based analysis, sentences that are routed through the KG branch are processed by a hybrid relation extraction pipeline to identify semantic triples. The rationale is to combine the strengths of an off-the-shelf Open Information Extraction (OpenIE) system [21] with those of a dependency-based extractor built on top of spaCy [24], aiming for robustness across different sentence types.

First, if an OpenIE backend is available, it is applied to S to obtain a set of candidate triples $\mathcal{T}_{\text{OIE}}$. These triples typically capture predicate-argument structures in a relatively model-agnostic way and are particularly effective on well-formed declarative sentences.

Second, the sentence is processed with a spaCy pipeline that performs tokenisation, part-of-speech tagging, dependency parsing, and named entity recognition. A set of hand-crafted patterns over the dependency tree is then used to extract additional triples $\mathcal{T}_{\text{spaCy}}$, for example by following `ROOT` verbs and their subjects and objects, by inspecting copular constructions, and by taking into account negations, adjectival modifiers, and prepositional links.

The two sets of triples are then merged into a unified candidate set

$$\mathcal{T} = \mathcal{T}_{\text{OIE}} \cup \mathcal{T}_{\text{spaCy}},$$

so that Phase 1 benefits both from the broad coverage of OpenIE and from the syntactic precision of dependency-based patterns. When the OpenIE backend is not available, the spaCy-based extractor alone provides a complete fallback, ensuring that the overall pipeline remains functional.

4.2.4 Triple Consolidation and Knowledge Graph Construction

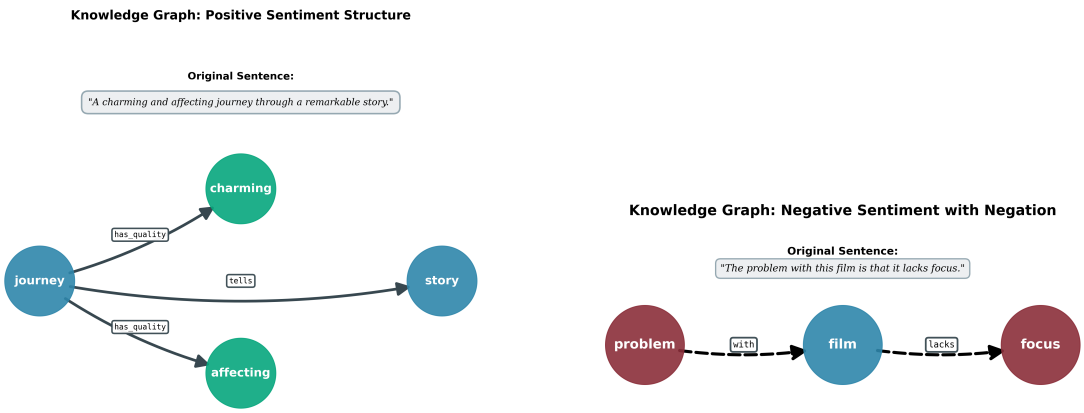
The merged set of triples $\mathcal{T}$ typically contains overlaps, paraphrases, and partially redundant variants of the same underlying relation. Before constructing the final knowledge graph, Phase 1 therefore applies a consolidation step that filters, groups, and merges triples that convey similar semantic content.

At a high level, this process involves:

- discarding low-quality triples that do not satisfy basic semantic or syntactic constraints (for instance, missing arguments or trivial predicates);
- normalising surface forms of entities and relations so that small lexical variations do not create separate nodes for the same concept;
- merging triples that are near-duplicates or that stand in an almost containment relationship (e.g., a longer triple that only extends a shorter one with a minor modifier).

The remaining triples are then used to construct a sentence-level knowledge graph $G(S)$. Each node in $G(S)$ corresponds to an entity (or, more generally, to a salient argument extracted from the sentence), and each directed edge encodes a semantic relation between two entities, labelled with the canonicalised predicate. The graph is implemented as an in-memory structure that can be eventually queried by the subsequent phases and, when needed, can be serialised to an RDF or OWL representation.

Besides building $G(S)$, the consolidated triples are also verbalised into a compact textual summary S_{KG}. This summary preserves the core entities and relations expressed in the original sentence while removing redundant details and surface-level variation. In practice, S_{KG} is constructed by linearly arranging the most salient triples according to simple templates and connective patterns, yielding a shorter and more structured sentence. The resulting graph and summary provide a compact and explicit view of the semantic content of S, which is particularly useful when the sentence is complex, includes multiple actors and events, or expresses contrasting opinions.



(a) Positive sentiment graph **(b)** Negative sentiment graph

Figure 4.2: Sentence-level knowledge graphs produced by Phase 1 for a positive (left) and a negative (right) sentiment-bearing sentence.

4.2.5 Example Knowledge Graphs

To illustrate the behaviour of Phase 1 on different sentence types, we show four representative examples of sentence-level knowledge graphs. Each graph is obtained by running the full Phase 1 pipeline on a single sentence and visualising the resulting entities and relations.

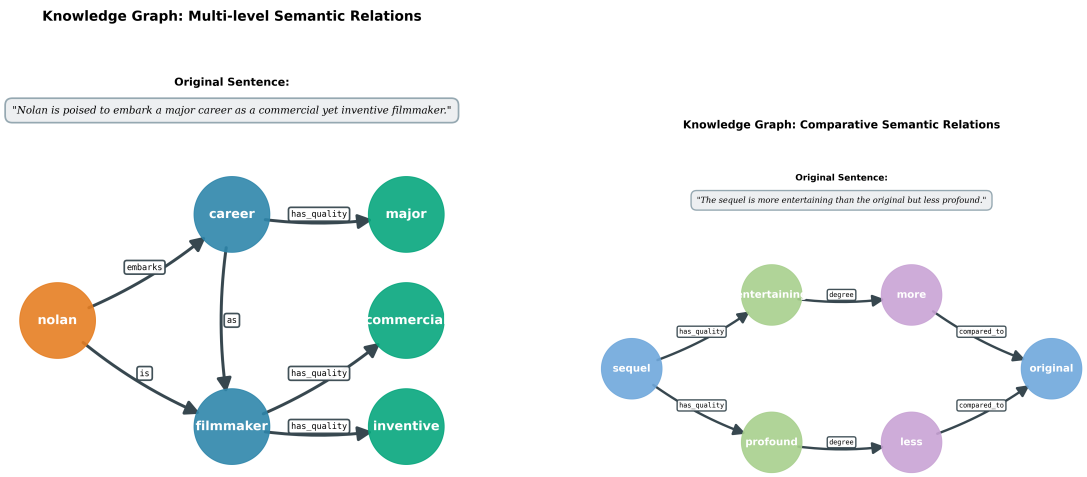
The first pair of examples focuses on polarity-bearing sentences, one expressing a clearly positive opinion and one expressing a negative opinion. In both cases, the knowledge graph highlights the main entities (for instance, the target of the opinion and the opinion holder) together with the evaluative predicates that connect them.

In the positive example, the graph makes explicit the favourable relation between the subject and the target entity, often via predicates or adjectives such as *“love”*, *“enjoyable”*, *“recommend”*, or similar. In the negative example, the graph instead captures negative predicates and modifiers, such as *“hate”*, *“boring”*, *“disappointing”*, or explicit negations. By comparing the two graphs side by side, it is immediate to see how the same Phase 1 pipeline encodes opposite semantic orientations using a similar structural pattern.

The second pair of examples targets more structurally complex sentences.

The first of these two is a syntactically rich sentence containing multiple clauses or events. The corresponding graph shows several interconnected triples, making the internal structure more transparent by exposing which entities participate in which relations and how the different clauses are linked. The last example is built from a comparative sentence in which two entities are explicitly contrasted (for example, one item being described as better or worse than another). Here, the graph encodes comparative relations as edges between the entities, with predicates

that capture the direction of the preference or the dimension along which the comparison is made.



(a) Complex sentence graph **(b)** Comparative sentence graph

Figure 4.3: Sentence-level knowledge graphs produced by Phase 1 for a complex sentence (left) and a comparative sentence (right).

Taken together, these four examples show that the same Phase 1 pipeline can handle diverse sentence structures and pragmatic functions, while always returning a coherent graph-based representation that feeds into the LLM-based encoder.

4.2.6 Configurable Parameters and Design Choices

Phase 1 exposes a set of configurable parameters that make it possible to trade off coverage, precision, and computational cost. The most relevant ones include:

- the entropy threshold H_θ that separates high- and low-entropy sentences, which influences how sentences are processed;
- the relative reliance on OpenIE versus spaCy-based patterns, which can be tuned depending on the availability and stability of the OpenIE backend;
- the criteria used to filter candidate triples (for instance, minimum confidence or minimum length), which control the density of the resulting knowledge graphs;
- the strength of the consolidation step, which determines how aggressively near-duplicate or overlapping triples are merged.

These parameters are fixed for the main set of experiments, but they can be adjusted in ablation studies to assess the impact of semantic preprocessing on the overall performance of the KG-LLM pipeline. In particular, the ability to disable Phase 1 entirely (i.e., to skip KG extraction and operate in a purely text-based mode) allows us to isolate the specific contribution of knowledge graphs to semantic robustness under noisy wireless conditions.

4.3 Phase 2 – LLM-Based Semantic Encoding

Phase 2 implements the LLM-based encoding block in Fig. 4.1 and the second phase of Algorithm 1. Its role is to compress the semantic content of the input sentence into a shorter, context-rich representation that can be transmitted over the wireless channel in either token-based or embedding-based form.

For each original sentence S, Phase 1 produces a structured object `ProcessedSentence` that includes both the original text and, when the KG branch is used, a condensed KG-driven summary S_{KG}. Phase 2 operates on a semantic input S_{sem} defined as follows:

- if the sentence has been processed through the KG branch, S_{sem} is derived from the KG-based summary S_{KG} (optionally combined with the original sentence S);
- if the sentence has bypassed KG extraction (e.g., high-entropy cases), S_{sem} coincides with the original sentence S.

In both cases, the encoder maps S_{sem} to a compressed semantic representation S_{enc} that is shorter than the original input but is designed to preserve the core meaning.

4.3.1 Encoder Architecture and Model Configuration

The encoder is implemented using transformer-based sequence-to-sequence models from the Hugging Face ecosystem [36]. In particular, the current framework supports the use of T5 [28] and BART [29] as encoder-decoder backbones. Both models provide an encoder that maps an input token sequence to a sequence of contextual embeddings, and a decoder that generates an output sequence conditioned on the encoder states.

In the T5 configuration, the input S_{sem} can be optionally prefixed with a task indicator such as *"summarize:"* to bias the model towards abstractive compression rather than generic generation. BART, on the other hand, does not require explicit task prefixes and is used directly in its sequence-to-sequence setting. The choice between T5 and BART is treated as a configuration parameter of the system and will be analysed in the evaluation chapter.

Both models share a common interface in Phase 2: they take tokenised text as input, perform an encoder forward pass, and then use a decoder with beam search to generate a compressed text sequence. The same code path handles either model, enabling fair comparisons between different architectures under identical channel and decoding conditions.

4.3.2 Encoding Pipeline and Token Merging

The single-sentence encoding pipeline in Phase 2 follows a sequence of operations that instantiates the high-level abstraction sketched in Algorithm 1:

1. **Tokenisation and input embeddings.** The semantic input S_{sem} is tokenised with the appropriate tokenizer (T5 or BART), producing a sequence of subword tokens of length n. These tokens are mapped to input embeddings and positional encodings, forming the encoder input tensor $H_{\text{input}} \in \mathbb{R}^{1\times n\times d}$.

2. **Encoder forward pass.** The encoder processes H_{input} and returns a contextual representation for each input position. This representation is mainly used for debugging and analysis, since the final compressed representation is derived from the decoder outputs.

3. **Decoder beam-search generation.** The decoder generates a shorter sequence of tokens by running a beam-search procedure with configurable parameters (e.g., number of beams, length penalty, minimum length ratio, and an `no_repeat_ngram_size` constraint). This yields a sequence of generated token IDs and the corresponding text, denoted by S_{enc}.

4. **Re-encoding of the generated text.** The truncated generated text S_{enc} is tokenised again and passed through the encoder to obtain a contextual representation $H_{\text{generated}} \in \mathbb{R}^{1\times m\times d}$ that is aligned with the tokens actually used for transmission.

5. **Subword merge into whole words.** Finally, a model-specific merge step is applied to group subword tokens into whole-word units. For example, the SentencePiece tokenizer used in T5 prefixes word-initial subwords with a special boundary marker, while the BART tokenizer encodes word boundaries through a dedicated leading-space flag. This produces a merged token sequence that is more robust to isolated symbol corruption and better aligned with human-interpretable words.

The outputs of this pipeline are collected in an `EncodedSequence` object, which contains:

- the merged token sequence `encoded_tokens`, used as payload for token-based transmission;

- the contextual representation $H_{\text{generated}}$, used as payload for embedding-based transmission;
- the generated text S_{enc}, useful for baseline comparisons at the bit or character level;
- metadata such as input and output lengths and compression ratios.

4.3.3 Embedding Extraction for Continuous Transmission

While `encoded_tokens` provides a natural discrete representation for token-based transmission, the tensor $H_{\text{generated}} \in \mathbb{R}^{1\times m\times d}$ offers a continuous embedding-based representation of the compressed sentence. In Phase 2, this tensor is prepared for the wireless channel module in a way that preserves the alignment between tokens and embeddings.

In the simplest case, $H_{\text{generated}}$ is kept as a three-dimensional array in which the first dimension corresponds to the batch (here, a single sentence), the second dimension indexes the generated tokens (m), and the third dimension indexes the embedding channels (d). When embedding-based transmission is selected, the channel module receives $H_{\text{generated}}$ and internally reshapes or flattens it into a one-dimensional stream of complex-valued symbols to be fed into the physical-layer model.

From the perspective of Phase 2, the important property is that each token in the compressed sentence is associated with a well-defined embedding vector that captures its contextual meaning. The subsequent Phase 3 will exploit this structure by feeding the received, possibly distorted embeddings directly into the decoder via cross-attention, without the need for an intermediate nearest-neighbour projection back to the vocabulary.

4.3.4 Configurable Parameters and Design Choices

Phase 2 exposes a number of hyperparameters that control the behaviour of the LLM-based encoder and make it possible to trade off compression, fluency, and robustness. The most relevant configuration options include:

- **Model choice and size.** The backbone can be set to T5 or BART, and, within each family, smaller or larger variants can be selected depending on the desired accuracy.
- **Use of task prefixes.** For T5, a task prefix such as *"summarize:"* can be enabled or disabled via a configuration flag, allowing a direct comparison between explicitly prompted summarisation and generic sequence generation.

In fact, T5 is a multitask transformer, designed to handle a wide range of NLP tasks, from summarization to translation[28]

- **Decoding strategy.** Beam search parameters such as the number of beams, length penalty, minimum and maximum output length, and the n-gram repetition constraint influence how concise or verbose the generated summaries are.

- **Merge strategy.** The rules used to merge subword tokens into whole words (e.g., markers considered, handling of punctuation) can be tuned to balance compatibility with the tokenizer and robustness to single-token corruption.

These parameters are fixed for the main set of experiments but can be varied in ablation studies to isolate the impact of different encoder configurations on compression effectiveness and downstream semantic reconstruction. Together with the Phase 1 configuration, they determine how much semantic content is passed to the channel and in which form, shaping the overall behaviour of the KG-LLM semantic communication pipeline.

4.4 Phase 3 – Semantic Decoding and Contextual Refinement

Phase 3 implements the receiver-side semantic decoding and refinement block in Fig. 4.1 and the third phase of Algorithm 1. Its role is to reconstruct a fluent and semantically faithful sentence from the noisy representation produced by the wireless channel, by combining three stages: a light pre-processing step to handle repetitions and corruption markers, an initial LLM-based reconstruction, and a BERT-based refinement procedure optionally guided by the knowledge graph extracted in Phase 1.

Depending on the selected transmission mode, Phase 3 operates either on a sequence of received tokens (token-based mode) or on a sequence of received contextual embeddings (embedding-based mode). In both cases, the decoder aims to recover a sentence that preserves the meaning of the original input, even when individual tokens or embedding components have been corrupted by channel noise or fading.

4.4.1 Module Overview and I/O Contract

From an implementation perspective, Phase 3 is encapsulated in a `semantic_decoding` module that exposes a unified interface for both transmission modes. The module receives the following inputs:

- in **token-based mode**, a sequence of received tokens together with a binary corruption mask that indicates which positions are considered unreliable by the channel model;
- in **embedding-based mode**, a tensor of received embeddings that mirrors the shape of the encoder output used in Phase 2;
- the original token sequence, used only for debugging and for the computation of accuracy metrics in the evaluation phase.

The main output of Phase 3 is a `DecodingResult` object that contains:

- the final reconstructed sentence S_{out};
- the intermediate reconstruction $S^{(0)}$ obtained by the initial LLM-based decoding;
- statistics on the refinement process, such as the number of tokens modified by BERT and the distribution of confidence scores;
- token-level confidence estimates and counts of anchor, corrected, and unknown tokens.

Phase 3 is organised internally into three steps: a pre-processing stage (Phase 0), an initial reconstruction stage (Phase 1), and a BERT-based refinement stage (Phase 2). This design mirrors the logical structure described in the system specification, while providing a clean separation between channel-aware decoding and semantic post-processing.

4.4.2 Pre-processing and Error Masking

The pre-processing step, referred to as Phase 0 in the implementation, prepares the received sequence for semantic decoding by handling spurious repetitions and explicit corruption markers inserted by the channel model.

In token-based mode, the input to this stage is a sequence of tokens and the associated corruption mask. Phase 0 performs two main operations:

- **Repetition detection and masking.** Consecutive duplicate tokens (such as *"It's It's"* or *"very very"*) are detected and treated as signs of possible corruption or generation artefacts. The second occurrence in each repetition is replaced by a special mask token, while the first occurrence is kept as an *anchor* token.

- **Integration of channel corruption information.** Positions marked as unreliable by the corruption mask are also replaced by mask tokens, unless they correspond to special placeholders introduced by the channel (e.g., generic error or unknown markers) or to punctuation. This ensures that the subsequent decoder explicitly sees gaps where the channel output is not trustworthy.

The result of Phase 0 is a *cleaned* token sequence in which reliable tokens act as anchors and unreliable or suspicious positions are explicitly masked. This sequence is then converted into a textual prompt to be fed to the LLM decoder in the next stage. In embedding-based mode, the pre-processing step is lighter: repetitions are optionally detected at the text level, but the core of the corruption is already encoded in the distortion of the embedding tensor received from the channel, which is passed directly to the decoder.

4.4.3 Initial LLM-Based Reconstruction

Phase 1 of the decoding process performs an initial reconstruction of the sentence using the same seq2seq LLM backbone employed in Phase 2 (T5 or BART). The objective is to exploit the model's contextual understanding to fill in masked positions and denoise the received representation before applying more targeted refinements.

In **token-based mode**, the cleaned token sequence produced by Phase 0 is converted into text, where anchor tokens appear as standard words and masked positions are represented by the appropriate mask symbol for the chosen model. This text is then fed to the decoder through the standard generation interface. The decoder uses beam search with the same family of hyperparameters as in Phase 2 (number of beams, length penalty, and an n-gram repetition constraint) to generate an initial reconstruction $S^{(0)}$ that attempts to restore a fluent sentence. When no masks are present (for instance, in low-noise conditions), the implementation can bypass the decoder and directly reuse the cleaned input to avoid unnecessary changes.

In **embedding-based mode**, the decoder operates directly on the received embeddings. The tensor of received embeddings is wrapped into an `encoder_outputs` structure and passed to the generation function as if it were the output of the encoder. If the embedding dimensionality does not match the expected model dimension, a lightweight linear projection layer is applied at runtime to align the shapes. The decoder then runs beam search with cross-attention over the noisy embeddings, producing the initial reconstruction $S^{(0)}$ without the need to map embeddings back to discrete tokens before decoding.

In both modes, Phase 1 outputs the tokenised form of $S^{(0)}$ together with the corresponding text. This representation already incorporates a significant amount

of semantic denoising, but may still contain local inconsistencies or artifacts due to strong channel noise or model uncertainty.

4.4.4 BERT-Based Refinement

The second decoding stage (Phase 2 in the implementation) refines the initial reconstruction $S^{(0)}$ using a BERT-based masked language model [16]. While the seq2seq decoder in Phase 1 operates at the sequence level and focuses on global fluency, BERT is used here as a local expert that proposes corrections for specific tokens that are likely to be erroneous.

The process starts by tokenising $S^{(0)}$ with the tokenizer associated with the chosen BERT model, which uses a WordPiece vocabulary. A set of heuristics is then applied to identify *suspicious* positions, including:

- tokens that originate from masked positions or explicit channel error markers;
- tokens that exhibit unusual repetition patterns or abnormal character sequences;
- tokens whose surrounding context strongly suggests an alternative choice (e.g., based on simple n-gram statistics or part-of-speech expectations).

For each suspicious position, the corresponding token is replaced by the BERT `[MASK]` symbol, and the masked sequence is fed to the BERT encoder. The model returns a probability distribution over the vocabulary for each masked position, from which a small set of candidate replacements is selected (typically the top-k tokens). A positional mapping between the original LLM tokens and the BERT tokens is maintained so that corrections can be consistently projected back onto the sentence produced in Phase 1.

By iterating this procedure over all suspicious positions, Phase 2 constructs a refined sentence $S^{(1)}$, along with token-level confidence scores that quantify how strongly BERT supports each proposed correction.

4.4.5 Configurable Parameters and Design Choices

Phase 3 exposes several configuration options that control the balance between conservative and aggressive refinement. The most relevant parameters include:

- **BERT model choice and size.** Different BERT variants (base vs. large, uncased vs. cased) can be selected depending on the desired trade-off between accuracy and computational cost.
- **Heuristics for suspicious token detection.** The rules used to flag tokens as suspicious (e.g., thresholds on repetition length, patterns of special characters,

or proximity to channel error markers) can be tuned to adjust the sensitivity of the refinement stage.

- **Top-k candidates and confidence thresholds.** The number of candidate replacements considered for each masked position and the minimum confidence required to apply a correction influence how many tokens are actually modified by BERT.
- **Enable/disable refinement stages.** BERT refinement can be enabled or disabled, allowing for ablation studies that isolate its contribution to overall performance.

As in the previous phases, these parameters are fixed for the main experiments presented in the evaluation chapter, but can be varied in controlled ablations to better understand the impact of semantic decoding and contextual refinement on robustness under different channel conditions.

Chapter 5

Experimental Evaluation

5.1 Evaluation Goals and Research Questions

The goal of this chapter is to experimentally assess whether the proposed KG-LLM pipeline can improve transmission efficiency and semantic robustness with respect to purely symbolic baselines, when messages are conveyed over a realistic radio channel affected by fading and additive noise. The experiments aim to quantitatively connect the "physical" layer (BER, SNR, modulation scheme) with the semantic layer (quality of the reconstructed text) and with the compression gain provided by the Knowledge Graph.

More specifically, we pursue four main objectives:

- **Robustness to channel noise:** evaluate how KG-LLM configurations behave as the SNR varies on a Rayleigh + AWGN channel, and whether they can maintain a more stable semantic quality than vanilla models.
- **Compression gain:** measure how much the KG-based preprocessing reduces the amount of transmitted information (characters, tokens, and bandwidth) compared to using the same T5 and BART architectures in a "vanilla" setting without Phase 1.
- **Token-mode vs embedding-mode:** study, for a given SNR, the differences between symbolic transmission based on token IDs and numerical transmission based on embeddings, also taking into account int8 quantization (1 byte per value) in the latter case.
- **Role of the decoder architecture (T5 vs BART):** compare the two models both in terms of compression capability and of semantic quality after decoding and BERT-based refinement.

Based on these objectives, we formulate the following research questions:

- **RQ1 - Robustness:** in the presence of a Rayleigh + AWGN channel, do KG-LLM configurations (KG-T5, KG-BART) maintain higher semantic metrics (BLEU, ROUGE-L, METEOR, BERTScore, sentence similarity) than their vanilla counterparts (NoKG-T5, NoKG-BART) at the same SNR?
- **RQ2 - Compression gain:** by how much does KG-based preprocessing reduce message length (in characters and tokens) and the effective bandwidth in token-mode and embedding-mode compared to vanilla models?
- **RQ3 - Transmission mode trade-off:** how do token-mode and embedding-mode compare, as the SNR varies, in terms of semantic quality and transmitted bandwidth? Is there an SNR regime in which one of the two modes is clearly preferable?
- **RQ4 - Model comparison:** for a given KG configuration and channel condition, which architecture (T5 vs BART) offers the best trade-off between compression capability and semantic reconstruction accuracy at the receiver?

5.2 Experimental Setup

5.2.1 Dataset

All experiments are conducted on the binary version of the *Stanford Sentiment Treebank* (SST-2) dataset. In our setup we work on a fixed validation split containing 872 sentences, each annotated with a global positive/negative sentiment label. Sentences are typically short, which makes this dataset well aligned with our scenario for three main reasons: (i) short sentences make it easier to isolate the impact of the channel and compression, without confounding factors due to very long texts; (ii) sentiment analysis requires preserving crucial semantic cues (polarity, intensity, negation); (iii) SST-2 is widely used in the NLP literature, which facilitates a qualitative comparison with related work.

For each combination of model (KG-T5, KG-BART, NoKG-T5, NoKG-BART), transmission mode (token-mode, embedding-mode), and SNR in the grid $\{2, 4, 6, 8, -10\}$ dB, we run the end-to-end pipeline on the full set of 872 sentences and compute:

- channel-level metrics (BER/TER or equivalent BER),
- textual and semantic metrics (BLEU, ROUGE-L, METEOR, BERTScore, SBERT similarity),
- compression and bandwidth statistics (characters, tokens, KB).

This guarantees a fair, one-to-one comparison across all configurations and directly links compression, channel robustness, and semantic quality on the same underlying data distribution.

5.2.2 Model and Pipeline Configurations

The experiments involve three main components: the encoder-decoder LLM, the refinement model, and the transmission pipeline.

Encoder-decoder LLMs. We consider two transformer-based encoder-decoder architectures:

- **T5:** we use the HuggingFace `t5-small` checkpoint, with a standard encoder-decoder architecture and hidden size 512. T5 is particularly suited to summarization and text-to-text tasks, making it a natural choice for Phase 2 semantic compression.
- **BART:** we use the `facebook/bart-base` checkpoint as a second encoder-decoder model. BART combines a denoising autoencoder pretraining objective with a Transformer encoder + decoder, and is often found to be strong on reconstruction-oriented tasks, complementing T5's more compressive behaviour.

Refinement model. For Phase 3 we employ a **BERT Masked Language Model**, e.g. the `bert-base-uncased` checkpoint. The refinement module is used to correct tokens that are suspected to be corrupted by the channel: suspicious positions are masked and BERT proposes high-probability replacements, with the option of leaving already reliable tokens unchanged.

Global pipeline configurations. We evaluate four main LLM configurations, obtained by toggling KG-based preprocessing and varying the underlying encoder-decoder:

- **KG ON (KG-LLM):** full pipeline with Phase 1 KG-based preprocessing (extraction and semantic compression), Phase 2 encoder-decoder (T5 or BART), and Phase 3 BERT refinement. In this mode, the channel input is a compressed, semantically grounded sentence.
- **KG OFF (vanilla):** LLM-only baseline without Phase 1, where the same encoder-decoder (T5 or BART) operates directly on the original sentence. Phase 3 remains active so that we can fairly isolate the contribution of KG-based preprocessing with respect to a purely LLM-based system.

We also vary the representation transmitted over the channel:

- **Token-based transmission:** the sequence of discrete tokens produced by the decoder is transmitted over the channel as 24-bit units (16 data bits + 8 CRC bits).
- **Embedding-based transmission:** the continuous decoder representations $H_{\text{generated}} \in \mathbb{R}^{m \times d}$ (before the output projection layer) are quantized to int8 and transmitted as flat vectors; the output text is decoded at the receiver side from the received embeddings (see also Section 5.2.3).

For each of the above settings we consider both a T5-based decoder and a BART-based decoder, paired with the corresponding encoder (KG-T5, KG-BART, NoKG-T5, NoKG-BART) in a symmetric way.

Decoding parameters are kept fixed across all conditions, for instance:

- beam search with `num_beams` = 4,
- a capped `max_length` to prevent overly verbose outputs,
- a `length_penalty` slightly below 1 to favour compact sequences,
- a `no_repeat_ngram_size` constraint to reduce spurious repetitions.

To reduce variance due to stochastic decoding and channel noise, each configuration is evaluated over multiple random seeds and/or multiple independent runs for each SNR value. Reported curves show averages across runs (and, where relevant, exhibit relatively small variance).

Taken together, these components define a family of configurations that can be organised along four evaluation axes, directly aligned with the research questions in Section 5.1:

- **KG-based preprocessing vs vanilla LLM (RQ2):** we compare KG-LLM (KG ON) against vanilla LLMs (KG OFF), always using the same underlying encoder-decoder and refinement stack. This allows us to quantify the impact of Phase 1 on character/token-level compression, effective bandwidth (in both token-mode and embedding-mode), and semantic quality.
- **Transmission mode: token vs embedding (RQ3):** for each model configuration (e.g., KG-T5, KG-BART, NoKG-T5, NoKG-BART) we evaluate both token-mode and embedding-mode. For each SNR value we compare the two modes in terms of semantic metrics and bandwidth, and identify SNR regimes where one representation becomes preferable over the other.

- **Decoder architecture: T5 vs BART (RQ4):** we contrast T5-based pipelines (KG-T5, NoKG-T5) with BART-based pipelines (KG-BART, NoKG-BART) under identical channel and SNR conditions. This highlights the trade-off between compression capability (where T5 tends to be more aggressive) and semantic reconstruction quality (where BART often achieves slightly higher scores), with and without KG-based preprocessing.
- **SNR sweep and channel conditions (RQ1):** all of the above configurations are evaluated over a common SNR grid, using the same Rayleigh+AWGN channel model, so that differences in performance can be attributed to the semantic pipeline rather than to changes in physical-layer conditions.

5.2.3 Wireless Channel Implementation

The channel simulation follows the theoretical model introduced in the wireless background chapter: a **Rayleigh flat-fading** channel with thermal **Additive White Gaussian Noise** (AWGN) at the equivalent bandwidth. On top of this, a simplified gNodeB/radio-stack abstraction can be superimposed to loosely mimic a 5G NR link.

Token-based transmission. In token-mode, the pipeline operates as follows:

- *Token encoding:* each token is mapped to a **16-bit ID**. A **CRC-8** is appended for single-token error detection, yielding **24 bits per token** (16 data bits + 8 redundancy bits).
- *Modulation:* bits are mapped to complex symbols according to a configurable `ModulationType`. In the main end-to-end experiments we use **QPSK**; the implementation also supports **BPSK** and **16-QAM** for alternative scenarios.
- *Channel:* the modulated symbol stream traverses a **Rayleigh fading** channel, optionally with multipath components, followed by **AWGN**. Fading coefficients are generated per coherence interval and normalized so that the average channel gain is unitary.
- *Output and internal metrics:* the receiver performs demodulation and bit-level detection, reconstructs token IDs, and checks the CRC. The channel module outputs:
  - the recovered token sequence `received_tokens`,
  - a boolean `corruption_mask` marking tokens declared as corrupted,
  - the **Bit Error Rate (BER)**,

  - the **Token Error Rate (TER)**,
  - the overall overhead due to headers, CRC, and framing.

  BER and TER are primarily used to verify that the simulation follows the expected BER-SNR behaviour of a realistic channel; they are not the final optimization target, but rather a realism constraint on top of which semantic evaluation is performed.

Embedding-based transmission. In embedding-mode, the system transmits continuous decoder representations:

- *Transmitted vector:* given the decoder output matrix $H_{\text{generated}} \in \mathbb{R}^{m \times d}$ (with m the sequence length and d the embedding dimension), we flatten it into a one-dimensional vector, apply **int8 quantization** (1 byte per value) to obtain a compact numerical representation, and map it to modulated symbols that traverse the same Rayleigh+AWGN channel. At the receiver, we perform fading equalization, dequantization, and reshape the vector back to the original $m \times d$ tensor before decoding.
- *Internal metrics:*
  - the **global mean squared error (MSE)** between transmitted and received embeddings,
  - the **average cosine similarity** between corresponding embedding vectors,
  - an **equivalent BER** indicator derived from the observed embedding distortion (formally defined in Section 5.3.1).

In all experiments, we use the same SNR grid $\{2, 4, 6, 8, 10\}$ dB, the same QPSK modulation scheme, and identical fading/multipath parameters across configurations, so that differences in performance can be attributed to the semantic pipeline rather than to changes in the underlying channel model.

5.3 Evaluation Metrics

This section introduces the metrics used to evaluate the proposed KG-LLM semantic communication system. We distinguish between channel-level metrics, which validate the physical-layer behaviour of the simulated link, and textual/semantic metrics, which quantify the quality of the reconstructed messages. In addition, we define compression and bandwidth metrics to capture the efficiency.

5.3.1 Channel-Level Metrics

Bit Error Rate (BER). The Bit Error Rate is defined as the ratio between the number of incorrectly decoded bits and the total number of transmitted bits:

$$\mathrm{BER} = \frac{N_{\text{bit, errors}}}{N_{\text{bit, total}}}. \tag{5.1}$$

In token-based transmission, we compute the BER by comparing, bit by bit, the transmitted sequence with the received sequence after demodulation and detection, before mapping bits back to token IDs. For each SNR value we record the observed BER and verify that it decreases with SNR in a way that is consistent with the theoretical Rayleigh+AWGN model presented in the wireless background chapter.

Token Error Rate (TER). The Token Error Rate is defined as the ratio between the number of incorrectly decoded tokens and the total number of transmitted tokens:

$$\mathrm{TER} = \frac{N_{\text{token, errors}}}{N_{\text{token, total}}}. \tag{5.2}$$

A token is considered erroneous if its reconstructed ID does not match the original ID (e.g., due to bit flips) or if the CRC-8 check fails. In the token-mode, TER is tightly coupled to BER and depends on the framing format (24 bits per token: 16 data bits + 8 CRC bits). A single bit flip can lead to a corrupted token ID or CRC failure, which may in turn result in an incorrect word or an `[UNK]` symbol at the text level.

Equivalent BER for embedding-mode. In embedding-based transmission there is no explicit notion of bits associated with discrete symbols. To obtain a channel-level indicator that is loosely comparable to BER, we define an *equivalent BER* by mapping the observed embedding distortion into a per-dimension error probability. In practice we consider the global mean squared error (MSE) between transmitted and received embeddings,

$$\mathrm{MSE} = \frac{1}{md} \sum_{i=1}^{m} \sum_{j=1}^{d} \left(h_{ij}^{(\mathrm{tx})} - h_{ij}^{(\mathrm{rx})} \right)^2, \tag{5.3}$$

and/or the loss in average cosine similarity, and normalize these quantities over the int8 quantization range to produce a scalar indicator that increases with distortion. This equivalent BER is *not* a physically accurate bit error probability; it is used exclusively as a qualitative proxy to visualise how distortion evolves with SNR in embedding-mode versus token-mode, and it does not enter as a direct design or optimization objective.

5.3.2 Textual and Semantic Metrics

To evaluate the textual and semantic quality of the reconstructed messages, we employ five standard metrics, computed using established Python libraries.

BLEU. The *Bilingual Evaluation Understudy* (BLEU) score measures n-gram overlap between a candidate sentence and a reference sentence, typically up to 4-grams. We use a sentence-level BLEU implementation with smoothing to avoid zero scores on short sentences. BLEU penalizes missing and spurious n-grams, making it a useful indicator of how much local lexical content is preserved after compression and channel distortion.[37]

ROUGE-L. ROUGE-L is based on the *Longest Common Subsequence* (LCS) between reference and candidate sentences. It computes a precision/recall-based F-measure derived from the LCS length. In contrast to BLEU, ROUGE-L is more sensitive to the preservation of the overall sentence structure and ordering of information, which is particularly relevant when assessing whether the high-level narrative of the message is maintained.[38]

METEOR. METEOR relies on word-level alignments that incorporate stemming and synonym matching (typically via WordNet). It combines precision, recall, and a fragmentation penalty into a single score. Because it explicitly accounts for morphological variants and synonyms, METEOR is well suited to semantic communication scenarios in which mild paraphrasing is acceptable as long as the underlying meaning is retained.[39]

BERTScore F1. BERTScore computes similarity between candidate and reference sentences using contextual embeddings from a pretrained BERT-like model. Each token in the candidate is matched to the most similar token in the reference in embedding space, and the resulting precision and recall are combined into an F1 score. Compared to surface metrics such as BLEU and ROUGE, BERTScore is more sensitive to semantic equivalence and robust to lexical variation, making it a natural choice to assess meaning preservation.[40]

Sentence-BERT similarity. Finally, we use a Sentence-BERT (SBERT) model to obtain a single embedding for each sentence and measure the cosine similarity between the original and reconstructed sentences. The resulting value, often reported as a percentage, captures the overall semantic proximity between the two sentences regardless of their surface form. In the context of semantic communication, this metric provides a direct indication of how close the reconstructed meaning is to the original one.[41]

5.3.3 Compression and Bandwidth Metrics

In addition to semantic quality, we measure how aggressively the proposed pipeline reduces the amount of information that needs to be transmitted over the channel. We consider both character-level and token-level compression, as well as an estimate of the effective bandwidth.

Character-level compression ratio. We define the character-level compression ratio as

$$\mathrm{CR}_{\mathrm{char}} = \frac{L_{\mathrm{tx,\ char}}}{L_{\mathrm{orig,\ char}}}, \tag{5.4}$$

where $L_{\mathrm{orig,\ char}}$ is the total number of characters in the original sentences and $L_{\mathrm{tx,\ char}}$ is the total number of characters in the compressed/summarized sentences produced by Phase 1-2. We compute $\mathrm{CR}_{\mathrm{char}}$ over the full set of 872 validation sentences, and compare KG-LLM configurations against vanilla baselines.

Token compression ratio. Similarly, we define the token-level compression ratio as

$$\mathrm{CR}_{\mathrm{token}} = \frac{N_{\mathrm{tx,\ token}}}{N_{\mathrm{orig,\ token}}}, \tag{5.5}$$

where $N_{\mathrm{orig,\ token}}$ is the total number of tokens in the original sentences and $N_{\mathrm{tx,\ token}}$ is the total number of tokens after Phase 1-2. Token counts are aggregated over the validation set from the pipeline logs (e.g., `total_input_tokens` and `total_output_tokens` recorded for T5 and BART), yielding global and average compression statistics.

Estimated bandwidth. We also estimate the amount of radio resources required to transmit the compressed representations.

- *Token-mode:* assuming 24 bits per token (16 data bits + 8 CRC bits) plus a fixed header overhead per experiment, the total bandwidth in kilobytes is approximated as

  $$\mathrm{KB}_{\mathrm{token}} = \frac{24 \cdot N_{\mathrm{token}} + \mathrm{overhead}}{8 \cdot 1024}. \tag{5.6}$$

  The corresponding plots report the total KB required by T5 and BART, with and without KG, over the full validation set.

- *Embedding-mode:* in embedding-based transmission we assume int8-quantized decoder embeddings (see Section 5.2.3). If $H_{\mathrm{generated}}$ has size $m \times d$, the bandwidth per *single transmission* is

  $$\mathrm{KB}_{\mathrm{emb}} = \frac{m \cdot d \cdot 8}{8 \cdot 1024} = \frac{m \cdot d}{1024}. \tag{5.7}$$

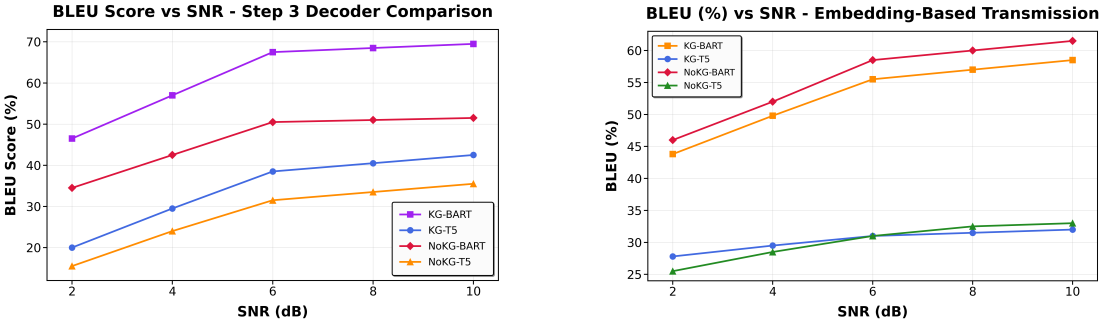
In the plots this quantity is multiplied by the number of sentences in the validation set and by the five SNR values used in the sweep, but the cost per message does not directly depend on the SNR; it is determined solely by the sequence length and the embedding dimensionality.

5.4 Results and Analysis

This section presents the quantitative results obtained from the end-to-end experiments. We first analyse how semantic quality varies with the SNR in token-based and embedding-based transmission, then we discuss compression and bandwidth figures derived from the full SST-2 validation set.

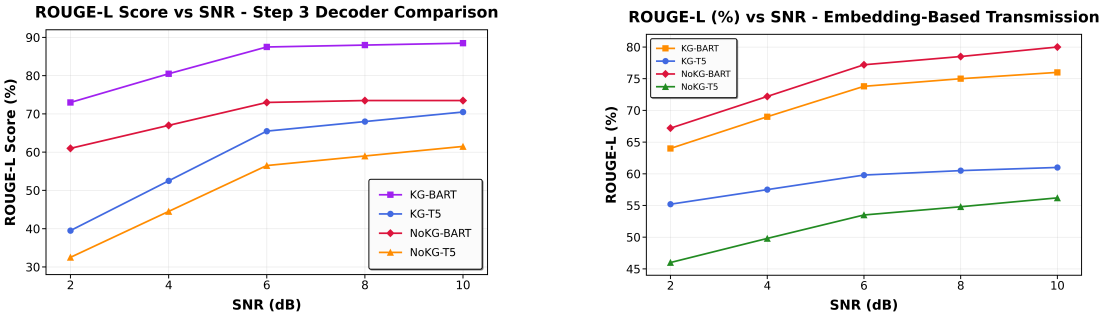
5.4.1 Semantic Quality vs SNR

Figures 5.1-5.5 report the evolution of BLEU, ROUGE-L, METEOR, BERTScoreF1 and SBERT sentence similarity as a function of the SNR for both token-based and embedding-based transmission. For each metric, the left subfigure corresponds to token-mode, while the right subfigure shows the same metric when transmitting decoder embeddings.



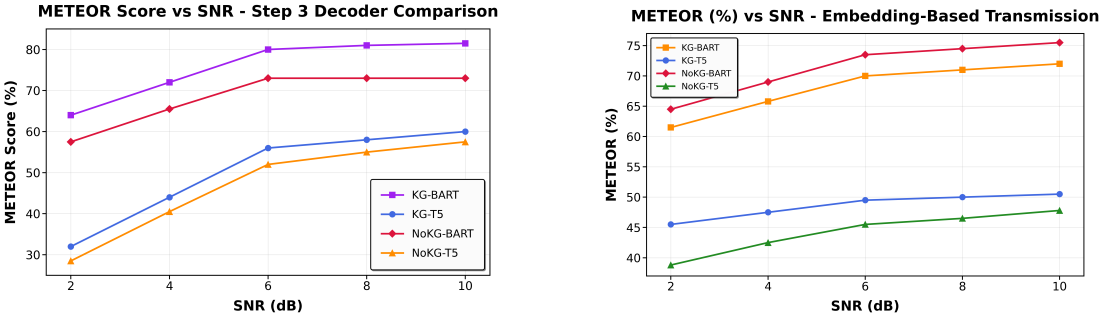
(a) Token-based transmission. **(b)** Embedding-based transmission.

Figure 5.1: BLEU score (%) vs SNR for token-based and embedding-based transmission.



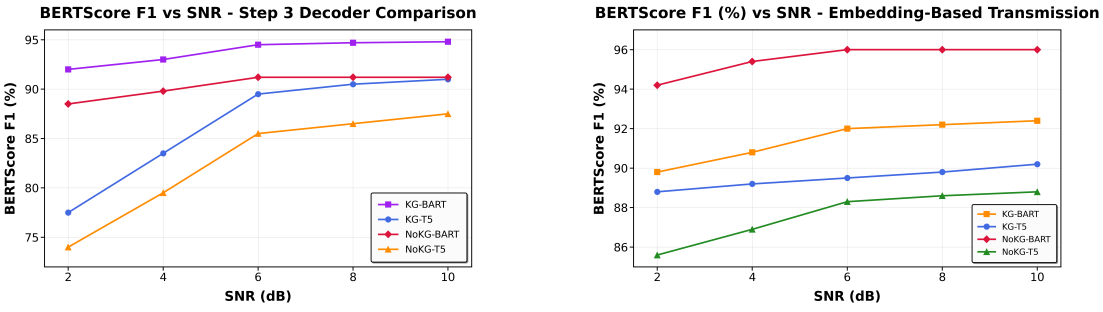
(a) Token-based transmission. **(b)** Embedding-based transmission.

Figure 5.2: ROUGE-L score (%) vs SNR for token-based and embedding-based transmission.



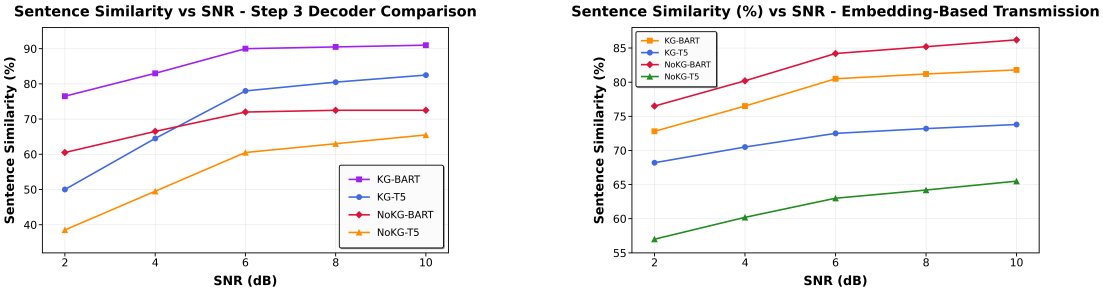
(a) Token-based transmission. **(b)** Embedding-based transmission.

Figure 5.3: METEOR score (%) vs SNR for token-based and embedding-based transmission.



(a) Token-based transmission. **(b)** Embedding-based transmission.

Figure 5.4: BERTScore F1 (%) vs SNR for token-based and embedding-based transmission.



(a) Token-based transmission.

(b) Embedding-based transmission.

Figure 5.5: Sentence-level cosine similarity (%) vs SNR for token-based and embedding-based transmission.

Across all metrics and configurations, semantic quality increases monotonically with SNR. At 2 dB the scores are clearly degraded, while at 8-10 dB the curves begin to saturate, indicating that residual channel errors become rare and that the dominant source of variability is the decoder itself rather than the wireless link.

In the **token-based** plots (left column), KG-based preprocessing has a consistently positive impact. For both T5 and BART, KG-LLM variants stay above their NoKG counterparts over the entire SNR range and for all five metrics. The gain is particularly visible at intermediate SNRs (4-6 dB), where the channel is still moderately noisy: KG-BART achieves markedly higher BLEU, ROUGE-L, and METEOR scores than NoKG-BART, and KG-T5 clearly outperforms NoKG-T5. Sentence-level similarity and BERTScore F1 show the same trend, confirming that the KG reduces the semantic damage produced by token errors in the channel.

Comparing **T5 and BART** in token-mode, BART achieves the highest semantic scores when combined with the KG. KG-BART is the top-performing configuration in almost all token-based plots, reflecting the strong reconstruction capabilities of BART's denoising pretraining. KG-T5 is more aggressive in compression (as discussed in Section 5.3.3), but its semantic scores remain slightly below KG-BART at a given SNR.

In the **embedding-based** plots (right column), the dependence on SNR is more gradual. Because small perturbations in the embedding space do not always translate into catastrophic word substitutions, the curves exhibit smoother slopes and no sharp cliff at low SNR. For T5, the KG-LLM variant retains a clear advantage over NoKG-T5 on contextual metrics such as BERTScore and sentence similarity at all SNR values, and improves METEOR and ROUGE-L as well. BLEU is the only metric where NoKG-T5 catches up or slightly overtakes KG-T5 at high SNR, reflecting the fact that BLEU is more sensitive to exact n-gram matches than to semantic equivalence.

For BART in embedding-mode the picture is more shaded. NoKG-BART often

achieves the highest scores among all configurations, particularly on BERTScore and sentence similarity. This suggests that when rich continuous representations are transmitted directly, BART can already exploit its internal semantic structure without requiring an explicit KG-based summarization step; overly aggressive compression may remove fine-grained details that would otherwise be preserved in the embedding stream. Even in this regime, however, KG-BART remains competitive and largely above T5-based pipelines at the same SNR.

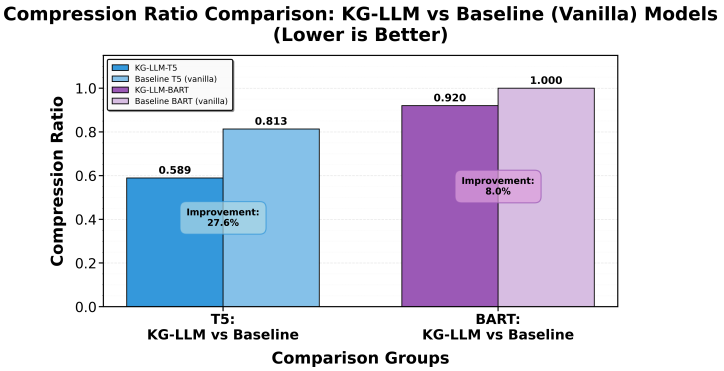
Overall, the semantic curves show that: (i) KG-based preprocessing systematically improves robustness for T5 in both token- and embedding-mode; (ii) for BART, KG provides a strong benefit in token-mode but a smaller (and sometimes negative) effect in embedding-mode; and (iii) embedding-based transmission tends to degrade more gracefully with SNR, while token-based transmission can achieve slightly higher peak scores when the channel is sufficiently clean.

5.4.2 Compression and Bandwidth Results

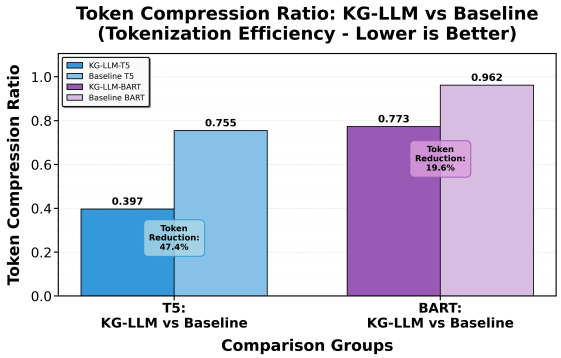
We now turn to compression and bandwidth figures, obtained by aggregating statistics over the full SST-2 validation set of 872 sentences. Character-level and token-level compression ratios are reported in Figure 5.6, while Figure 5.7 summarises the estimated transmission bandwidth in token-based and embedding-based modes.

At the **character level**, KG-LLM achieves substantial compression compared to vanilla models. For T5, the compression ratio drops from approximately 0.81 (baseline) to 0.59 with KG-based preprocessing, corresponding to a reduction of about 28% in the number of characters transmitted. For BART the effect is milder but still noticeable, with the ratio going from 1.00 (no compression) to around 0.92, i.e., roughly an 8% reduction.

The effect is even more pronounced at the **token level**. For T5, the token compression ratio decreases from about 0.76 in the baseline to 0.40 with KG-LLM, meaning that the number of tokens to be transmitted is almost halved (a reduction of roughly 47%). For BART, the ratio drops from approximately 0.96 to 0.77, corresponding to a token reduction close to 20%. These results confirm that Phase 1 significantly shortens the sequences seen by the encoder-decoder, especially for a summarization-oriented model such as T5.

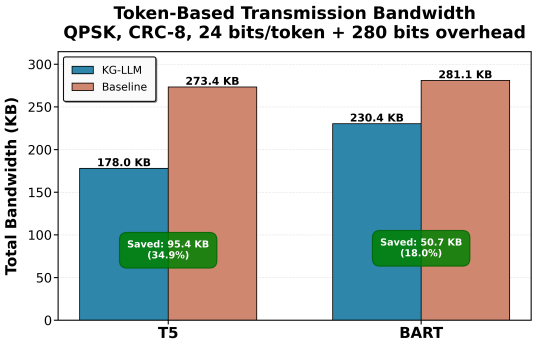


(a) Character-level compression ratio.

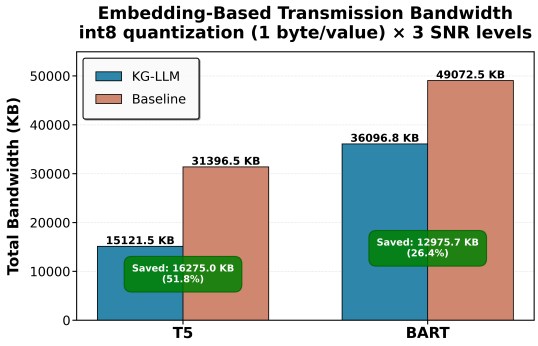


(b) Token-level compression ratio.

Figure 5.6: Compression ratio of KG-LLM vs vanilla baselines on the SST-2 validation set (lower is better).



(a) Token-based transmission bandwidth.



(b) Embedding-based transmission bandwidth.

Figure 5.7: Estimated total bandwidth (KB) required by KG-LLM and vanilla models in token-based and embedding-based transmission.

These compression gains translate into tangible **bandwidth savings** at the physical layer. In token-mode (QPSK, 24 bits/token plus protocol overhead), the total bandwidth over the validation set decreases from about 273.4 KB to 178.0 KB for T5, and from roughly 281.1 KB to 230.4 KB for BART. This corresponds to a bandwidth reduction of approximately one third for T5 and nearly one fifth for BART.

In embedding-mode (int8 quantization, 1 byte per embedding value), the savings are larger in absolute terms. For T5, KG-LLM reduces the total transmitted volume from about 31.4 MB to 15.1 MB, i.e., slightly more than 50% fewer kilobytes. For BART the reduction is from roughly 49.1 MB to 36.1 MB, yielding savings on the order of 25-30%. Since these quantities scale linearly with the number of messages, the impact on aggregate airtime and spectral efficiency becomes even more relevant in large-scale deployments.

Taken together, the semantic curves and the compression/bandwidth plots indicate that KG-based preprocessing can substantially reduce the amount of information that must be carried by the wireless link while maintaining semantic quality, especially for T5-based pipelines and for scenarios where token-based transmission is used at moderate SNR values. These trends will be further interpreted in the discussion section.

5.5 Discussion

This section revisits the research questions introduced in Section 5.1 in the light of the results presented in Section 5.4, and discusses their implications for semantic communication in bandwidth-constrained, low-SNR edge scenarios.

RQ1 - Robustness to channel noise

The semantic curves in Figures 5.1-5.5 show that all configurations benefit from increasing SNR, but KG-LLM pipelines are consistently more robust than their vanilla counterparts in several regimes. In token-based transmission, KG-T5 and KG-BART dominate NoKG-T5 and NoKG-BART across the entire SNR sweep for all five metrics, with the largest gaps appearing at intermediate SNR values (4-6 dB). This suggests that KG-based preprocessing makes the transmitted sequences intrinsically more redundant at the semantic level, so that occasional token errors are less likely to flip the overall meaning of the message.

In embedding-based transmission the dependence on SNR is smoother and the effect of the KG is more model-dependent. For T5, KG-LLM clearly improves BERTScore and sentence similarity for all SNRs, while for BART the NoKG variant can match or slightly outperform KG-BART on some metrics at high SNR. Overall, the results support the view that KG-guided compression improves semantic

robustness in the most challenging regime (moderate SNR with non negligible distortion), especially for models such as T5 that are trained for summarisation.

RQ2 - Compression gain

The compression and bandwidth plots in Figures 5.6-5.7 confirm that KG-based preprocessing significantly reduces the amount of information that must be transmitted.

At the character level, KG-T5 lowers the compression ratio from approximately 0.81 to 0.59, corresponding to a reduction of about 27.6% in the number of characters. KG-BART still achieves a non-negligible improvement, from roughly 1.00 to 0.92 (about 8%). Token-level compression is even more pronounced: KG-T5 almost halves the number of tokens to be transmitted (ratio ≈ 0.40 vs 0.76, i.e. a reduction around 47.4%), while KG-BART reduces tokens by about 19.7%.

These gains translate into concrete savings in radio resources. In token-based transmission (QPSK, 24 bits/token + overhead) the total bandwidth over the 872-sentence validation set drops from 273.4 KB to 178.0 KB for T5, and from 281.1 KB to 230.4 KB for BART. In embedding-mode (int8 quantisation) KG-T5 reduces the transmitted volume from 31 396.5 KB to 15 121.5 KB (about 51.8% fewer kilobytes), while KG-BART goes from 49 072.5 KB to 36 096.8 KB (roughly a 25-30% reduction). Since these quantities scale linearly with the number of messages, even moderate per-sentence savings can become substantial in large deployments.

RQ3 - Token-mode vs embedding-mode

Comparing the left and right subfigures in Figures 5.1-5.5, two main patterns emerge.

First, *token-mode* exhibits a sharper dependence on SNR: when the BER is high (2 dB), a few bit flips are enough to corrupt entire tokens, leading to noticeable drops in BLEU and ROUGE-L; as the SNR increases, TER quickly decreases and semantic scores saturate. In this regime KG-BART is the strongest configuration, especially at 6-10 dB.

Second, *embedding-mode* degrades more gracefully with SNR. Because small perturbations in the embedding space do not necessarily change the most likely decoded word, the curves are smoother and low-SNR performance is less catastrophic. At high SNR, token-mode can reach slightly higher peak scores, while embedding-mode offers larger bandwidth savings thanks to int8 quantisation and is less sensitive to discrete symbol errors.

The choice between token-based and embedding-based transmission is therefore scenario-dependent. In links that routinely operate at moderate-to-high SNR,

token-mode with KG-BART or KG-T5 provides excellent semantic quality with moderate compression. In harsher conditions, or when very aggressive quantisation is needed, embedding-mode becomes attractive because it avoids hard symbol errors and can still deliver acceptable semantic similarity while reducing the number of bytes per message.

RQ4 - T5 vs BART

The comparison between T5 and BART highlights a clear division of roles. KG-T5 is consistently the most *compressive* configuration: it produces the shortest summaries and achieves the lowest character and token compression ratios, which directly translate into the largest bandwidth reductions. This behaviour is aligned with its pretraining on summarisation and other text-to-text tasks.

KG-BART, on the other hand, tends to be the most *accurate* decoder in terms of semantic metrics, especially in token-mode. Its denoising autoencoder pretraining allows it to reconstruct fluent, faithful sentences even when the input has been affected by token errors or by the aggressive Phase 1 compression.

These observations suggest a potential future extension in which T5 and BART are combined in a hybrid pipeline: a T5-based encoder performing strong semantic compression, followed by a BART-based decoder (possibly with KG-aware prompts) specialised in faithful reconstruction and refinement. Such a configuration could exploit the best of both worlds, further pushing the trade-off between compression and semantic robustness.

Chapter 6

Conclusions

This thesis set out to investigate whether a hybrid semantic communication architecture, combining knowledge graphs (KGs) and large language models (LLMs), can transmit the *meaning* of short textual messages over noisy wireless channels more efficiently and robustly than conventional symbol-based approaches. The central goal was to move from bit-level reliability towards meaning-oriented performance, in line with the needs of bandwidth-constrained edge and IoT scenarios.

Within this perspective, the KG-LLM framework combines symbolic and neural components in a complementary way. KG-based preprocessing reduces redundancy and exposes the core semantic structure of each message, highlighting the entities and relations that carry most of the meaning. LLMs then exploit this condensed, structured input to produce compact semantic representations and to reconstruct fluent text at the receiver, relying on their contextualisation capabilities to restore details that are not explicitly transmitted. End-to-end evaluation under noisy-channel conditions, based on task-agnostic measures of semantic similarity, provides a direct view of how well the reconstructed messages preserve the intended meaning.

The experimental results in Chapter 5 show that this cooperation between KGs and LLMs translates into tangible gains. KG-aware configurations generally preserve or improve semantic similarity compared to LLM-only baselines, especially at low-to-medium SNR values, while at the same time reducing the amount of information that needs to be sent. Token-based transmission proves highly efficient and accurate when channel conditions are favourable, but degrades more abruptly as SNR decreases, reflecting its sensitivity to symbol errors. Embedding-based transmission, instead, offers a smoother degradation profile thanks to the continuous nature of the representations, at the cost of different bandwidth and implementation trade-offs. Across both modes, T5 tends to behave as a strong semantic compressor, whereas BART often delivers slightly better reconstruction quality, particularly when guided by KG-derived summaries. Overall, the empirical evidence supports the viability of KG-LLM semantic communication as a candidate solution for

meaning-oriented, bandwidth-efficient transmission in noisy environments.

At the same time, this work represents only an initial step. The current evaluation is limited to short English sentences from a single benchmark and to relatively compact pretrained models, tested under a simplified channel setting. Future research can build on these foundations by extending the framework to longer texts, dialogues, multilingual traffic and more diverse tasks; and by integrating the semantic pipeline into full network simulators or experimental testbeds to assess end-to-end behaviour under realistic protocol stacks and multiuser operation. Within these limitations, the thesis demonstrates that explicitly combining symbolic knowledge and neural language models can significantly improve efficiency, resilience, and semantic similarity in end-to-end communication, and points towards a broader transition from bit-oriented wireless systems to architectures that explicitly optimise for the preservation of meaning.

Bibliography

[1] C.E. Shannon and W. Weaver. *The Mathematical Theory of Communication*. Urbana, IL: University of Illinois Press, 1949 (cit. on pp. 1, 2, 16, 45).

[2] G. Xin, P. Fan, and K.B. Letaief. «Semantic Communication: A Survey of Its Theoretical Development». In: *Entropy* 26.2 (2024), p. 102. DOI: 10.3390/e26020102. URL: https://www.mdpi.com/1099-4300/26/2/102 (cit. on pp. 2, 3, 15, 16).

[3] T. Han, Q. Yang, Z. Shi, et al. *Semantic-preserved Communication System for Highly Efficient Speech Transmission*. 2022. arXiv: 2205.12727 [eess.AS]. URL: https://arxiv.org/abs/2205.12727 (cit. on pp. 6, 7).

[4] M. Lokumarambage, V. Gowrisetty, H. Rezaei, et al. «Wireless End-to-End Image Transmission System using Semantic Communications». In: *Proc. IEEE International Conference on Communications*. 2023. URL: https://arxiv.org/abs/2302.13721 (cit. on p. 7).

[5] Z. Wang, L. Zou, S. Wei, et al. «Large-Language-Model-Enabled Text Semantic Communication Systems». In: *Applied Sciences* 15.13 (2025), p. 7227. DOI: 10.3390/app15137227 (cit. on p. 8).

[6] M. Chen, Z. Sun, X. He, et al. «LLM-Based Semantic Communication: The Way From Task-Originated to General». In: *IEEE Wireless Communications Letters* 14.10 (2025), pp. 3029–3033. DOI: 10.1109/LWC.2025.3583053 (cit. on p. 8).

[7] A. Kalita. *Large Language Models (LLMs) for Semantic Communication in Edge-based IoT Networks*. 2024. arXiv: 2407.20970 [cs.NI]. URL: https://arxiv.org/abs/2407.20970 (cit. on p. 9).

[8] S. Salehi, M. Erol-Kantarci, and D. Niyato. *LLM-Enabled Data Transmission in End-to-End Semantic Communication*. 2025. arXiv: 2504.07431 [cs.NI]. URL: https://arxiv.org/abs/2504.07431 (cit. on pp. 9, 13, 34).

[9] S. Jiang, Y. Liu, Y. Zhang, et al. «Reliable Semantic Communication System Enabled by Knowledge Graph». In: *Entropy* 24.6 (2022), p. 846. DOI: 10.3390/e24060846 (cit. on p. 10).

[10] C. Guo, J. Liu, W. Gao, et al. «A Large Language Model Driven Knowledge Graph Construction Scheme for Semantic Communication». In: *Applied Sciences* 15.8 (2025), p. 4575. DOI: 10.3390/app15084575 (cit. on pp. 10, 11).

[11] B. Wang, R. Li, J. Zhu, et al. *Knowledge Enhanced Semantic Communication Receiver*. 2023. arXiv: 2302.07727 [cs.CL]. URL: https://arxiv.org/abs/2302.07727 (cit. on pp. 10, 12).

[12] C. Liang, Y. Sun, D. Nyato, and M.A. Imran. «Knowledge Graph Fusion Based Semantic Communication Framework». In: *IEEE Transactions on Mobile Computing* 24.11 (2025), pp. 11416–11429. DOI: 10.1109/TMC.2025.3583605 (cit. on pp. 12, 13, 18).

[13] O. Goldreich, B. Juba, and M. Sudan. «A Theory of Goal-Oriented Communication». In: *Journal of the ACM* 59.2 (2012), 8:1–8:65. DOI: 10.1145/2160158.2160161 (cit. on p. 15).

[14] D. Wheeler and B. Natarajan. «Engineering Semantic Communication: A Survey». In: *IEEE Access* 11 (2023), pp. 13965–13995. ISSN: 2169-3536. DOI: 10.1109/access.2023.3243065. URL: http://dx.doi.org/10.1109/ACCESS.2023.3243065 (cit. on p. 17).

[15] T. Mikolov, I. Sutskever, K. Chen, et al. *Distributed Representations of Words and Phrases and their Compositionality*. 2013. arXiv: 1310.4546 [cs.CL]. URL: https://arxiv.org/abs/1310.4546 (cit. on p. 18).

[16] J. Devlin, M. Chang, K. Lee, and K. Toutanova. «BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding». In: 2019. arXiv: 1810.04805 [cs.CL]. URL: https://arxiv.org/abs/1810.04805 (cit. on pp. 18, 34, 36, 55).

[17] A. Hogan, E. Blomqvist, M. Cochez, C. D'Amato, et al. «Knowledge Graphs». In: *ACM Computing Surveys* 54.4 (2021), pp. 1–37. DOI: 10.1145/3447772 (cit. on pp. 18, 20).

[18] P. Schneider, T. Schopf, J. Vladika, et al. «A Decade of Knowledge Graphs in Natural Language Processing: A Survey». In: *Proceedings of the 2nd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 12th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Association for Computational Linguistics, 2022, pp. 601–614. DOI: 10.18653/v1/2022.aacl-main.46 (cit. on p. 18).

[19] O. Etzioni, M. Banko, S. Soderland, and D.S. Weld. «Open information extraction from the web». In: *Commun. ACM* 51.12 (Dec. 2008), pp. 68–74. ISSN: 0001-0782. DOI: 10.1145/1409360.1409378. URL: https://doi.org/10.1145/1409360.1409378 (cit. on p. 21).

[20] A. Yates, M. Banko, M. Broadhead, et al. «TextRunner: Open Information Extraction on the Web». In: *Proceedings of Human Language Technologies: The Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL-HLT)*. Ed. by Bob Carpenter, Amanda Stent, and Jason D. Williams. Rochester, New York, USA: Association for Computational Linguistics, Apr. 2007, pp. 25–26. URL: https://aclanthology.org/N07-4013/ (cit. on p. 21).

[21] O. Etzioni, A. Fader, J. Christensen, et al. «Open information extraction: the second generation». In: *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence - Volume Volume One*. IJCAI'11. Barcelona, Catalonia, Spain: AAAI Press, 2011, pp. 3–10. ISBN: 9781577355137 (cit. on pp. 21, 45).

[22] G. Angeli, M. J. J. Premkumar, and C. D. Manning. «Leveraging Linguistic Structure for Open Domain Information Extraction». In: *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Ed. by Chengqing Zong and Michael Strube. Beijing, China: Association for Computational Linguistics, July 2015, pp. 344–354. DOI: 10.3115/v1/P15-1034. URL: https://aclanthology.org/P15-1034/ (cit. on p. 22).

[23] C. Niklaus, M. Cetto, A. Freitas, and S. Handschuh. «A Survey on Open Information Extraction». In: *Proceedings of the 27th International Conference on Computational Linguistics*. Ed. by Emily M. Bender, Leon Derczynski, and Pierre Isabelle. Santa Fe, New Mexico, USA: Association for Computational Linguistics, Aug. 2018, pp. 3866–3878. URL: https://aclanthology.org/C18-1326/ (cit. on p. 22).

[24] M. Honnibal, I. Montani, S. Van Landeghem, and A. Boyd. «spaCy: Industrial-strength Natural Language Processing in Python». In: (2020). DOI: 10.5281/zenodo.1212303 (cit. on pp. 22, 45).

[25] M. Honnibal and I. Montani. *spaCy 101: Everything You Need to Know*. https://spacy.io/usage/spacy-101. Accessed: 2025-11-18. 2020 (cit. on p. 22).

[26] A. Vaswani, N. Shazeer, N. Parmar, et al. «Attention Is All You Need». In: *Advances in Neural Information Processing Systems*. Vol. 30. Curran Associates, Inc., 2017. URL: https://papers.nips.cc/paper/7181-attention-is-all-you-need (cit. on pp. 24, 25, 27).

[27] I. Sutskever, O. Vinyals, and Q. V. Le. «Sequence to Sequence Learning with Neural Networks». In: *Advances in Neural Information Processing Systems*. Vol. 27. Curran Associates, Inc., 2014. URL: https://papers.nips.cc/paper/5346-sequence-to-sequence-learning-with-neural-networks (cit. on p. 27).

[28] C. Raffel, N. Shazeer, A. Roberts, et al. «Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer». In: *Journal of Machine Learning Research* 21.140 (2020), pp. 1–67. URL: https://jmlr.org/papers/v21/20-074.html (cit. on pp. 27, 35, 49, 52).

[29] M. Lewis, Y. Liu, N. Goyal, et al. «BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension». In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. 2020, pp. 7871–7880. DOI: 10.18653/v1/2020.acl-main.703. URL: https://aclanthology.org/2020.acl-main.703 (cit. on pp. 27, 35, 49).

[30] S.W. Tan, C. Lee, K. M. Lim, et al. «QARR-FSQA: Question-Answer Replacement and Removal Pretraining Framework for Few-Shot Question Answering». In: *IEEE Access* 12 (2024), pp. 159280–159295. DOI: 10.1109/ACCESS.2024.3487581 (cit. on p. 27).

[31] S. Swain. *BART Model for Text Auto Completion in NLP. GeeksforGeeks*. Last accessed: 19 Nov. 2025. 2025. URL: https://www.geeksforgeeks.org/artificial-intelligence/bart-model-for-text-auto-completion-in-nlp (cit. on p. 28).

[32] A. Goldsmith. *Wireless Communications*. Cambridge, UK: Cambridge University Press, 2005 (cit. on p. 31).

[33] D. Tse and P. Viswanath. *Fundamentals of Wireless Communication*. Cambridge, UK: Cambridge University Press, 2005 (cit. on p. 31).

[34] *Add white Gaussian noise to signal. MathWorks Documentation*, awgn function. Accessed: 2025. URL: https://www.mathworks.com/help/comm/ref/awgn.html (cit. on p. 31).

[35] Turing. *How the BERT NLP Optimization Model Works. Turing Blog*. 2025. URL: https://www.turing.com/kb/how-bert-nlp-optimization-model-works (cit. on p. 36).

[36] T. Wolf, L. Debut, V. Sanh, et al. «Transformers: State-of-the-Art Natural Language Processing». In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. Online: Association for Computational Linguistics, Oct. 2020, pp. 38–45. DOI: 10.18653/v1/2020.emnlp-demos.6. URL: https://aclanthology.org/2020.emnlp-demos.6/ (cit. on p. 49).

[37] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. «BLEU: a Method for Automatic Evaluation of Machine Translation». In: *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics. 2002, pp. 311–318. DOI: 10.3115/1073083.1073135 (cit. on p. 64).

[38] Chin-Yew Lin. «ROUGE: A Package for Automatic Evaluation of Summaries». In: *Text Summarization Branches Out*. Barcelona, Spain: Association for Computational Linguistics, July 2004, pp. 74–81. URL: https://aclanthology.org/W04-1013 (cit. on p. 64).

[39] Satanjeev Banerjee and Alon Lavie. «METEOR: An Automatic Metric for MT Evaluation with Improved Correlation with Human Judgments». In: *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*. Ann Arbor, Michigan: Association for Computational Linguistics, 2005, pp. 65–72. URL: https://aclanthology.org/W05-0909 (cit. on p. 64).

[40] Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. «BERTScore: Evaluating Text Generation with BERT». In: *CoRR* abs/1904.09675 (2019). arXiv: 1904.09675. URL: http://arxiv.org/abs/1904.09675 (cit. on p. 64).

[41] Nils Reimers and Iryna Gurevych. «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)*. Hong Kong, China: Association for Computational Linguistics, 2019, pp. 3980–3990. DOI: 10.18653/v1/D19-1410. URL: https://doi.org/10.18653/v1/D19-1410 (cit. on p. 64).