SCIENCE CHINA
Information Sciences

• RESEARCH PAPER •

July 2023, Vol. 66 172302:1–172302:17
https://doi.org/10.1007/s11432-022-3624-4

Knowledge-enhanced semantic communication system with OFDM transmissions

Xiaodong XU[1,2], Huachao XIONG[1], Yining WANG[1], Yue CHE[1], Shujun HAN[1*], Bizhu WANG[1] & Ping ZHANG[1,2]

[1]*State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing* 100876*, China;*
[2]*Department of Broadband Communication, Peng Cheng Laboratory, Shenzhen* 518055*, China*

Received 1 July 2022/Revised 27 September 2022/Accepted 16 November 2022/Published online 21 June 2023

Abstract As a promising technology to enable effective multi-modal transmission over wireless channels, semantic communication has attracted a lot of attention from academics and industries. Different from Shannon's information theory, based on common background knowledge provided by the knowledge base, the goal of semantic communication is transmitting intended useful information from the transmitter and recovered by the receiver at the semantic level. However, the existing studies on semantic communication rarely emphasize the essence and the usage of the knowledge base. In this paper, we propose a knowledge-enhanced semantic communication (KESC) system, where the knowledge base is cloud-edge-device collaborative cached. To solve the problem that float-type symbols are difficult to transmit directly through a radio frequency (RF) system, we adopt orthogonal frequency division multiplexing (OFDM) to transmit semantic vectors directly without some traditional signal processing techniques in semantic information transmission, and the semantic pilot is designed to assist semantic reception. Furthermore, we formulate a multi-encoder transformer based neural network model for the KESC system to support text transmission (KESC-T), where the decoder is implemented with a knowledge graph to enhance the performance of semantic decoding. Besides, we define knowledge-enhanced efficiency (KEE) to measure the gain in semantic recovery accuracy brought by per unit of knowledge. Simulation results demonstrate that the recovery accuracy of our proposed KESC outperforms the compared scheme, especially in low signal-to-noise ratio (SNR) or resource-constrained scenarios.

Keywords semantic communication, knowledge graph, OFDM, transformer, deep learning

Citation Xu X D, Xiong H C, Wang Y N, et al. Knowledge-enhanced semantic communication system with OFDM transmissions. Sci China Inf Sci, 2023, 66(7): 172302, https://doi.org/10.1007/s11432-022-3624-4

1 Introduction

Towards the 2030s, virtual and real worlds will be blended seamlessly with the support of the 6th generation communication system (6G) and artificial intelligence technologies. Viewed as an enabler and beneficiary of ubiquitous intelligent services, 6G emphasizes the effectiveness and sustainability of information [1,2]. However, the evolution toward beyond-5G (B5G) still has challenges in effectively handling the ever-increasing requirements brought by revolutionary services, like autonomous driving, metaverse, or holographic communication. Going beyond the Shannon paradigm of achieving bit-level error-free transmission, semantic communication focuses on "what" to transmit at the semantic level. With the development of edge intelligence, semantic communication is viewed as a promising technology for artificial Internet of Things (AIoT) devices with capabilities of signal processing and computation functionality, attracting lots of attention from academics and industrials [3].

Semantic communication was first formulated by Weaver and Shannon in 1949, then extended into universal semantic communication [4] in 2008 and goal-oriented communication [5] in 2012. It is expected to break through the transmission bottleneck of the classical symbol-level communication system and provide new solutions for 6G. Traditional symbol-level communication requires that the decoded symbols at

* Corresponding author (email: hanshujun@bupt.edu.cn)

the receiver are strictly consistent with the encoded symbols at the transmitter. While semantic communication conveys the meanings intended by a transmitter, it only requires that the semantic information recovered by the receiver matches the semantic information sent by the transmitter [6,7]. Based on deep learning technologies, semantic processing can further compress the transmission content and relieve the transmission pressures of networks. Meanwhile, some metrics of reliability performance for useful information transmission are defined as bilingual evaluation understudy (BLEU) [8], sentence similarity [7], and word error rate (WER) [9], in order to measure the error tolerance of information transmission.

Focusing on the design of transmitter, channel, and receiver, different schemes have been proposed in many researches [7, 10–15]. In these studies, semantic communication is designed as a point-to-point communication process between the transmitter and receiver, and neural network models are deployed in the transmitter and receiver for semantic extraction and semantic recovery [7, 10–13]. Since the performance of neural networks is related to the training data, training methods, and training equipment, the performance of the studied semantic communication systems in [7,10–13] will degenerate when applied the intelligence devices with limited computing resources.

To further improve the performance of semantic communication, deploying a unified background knowledge base on both transmitter and receiver is proposed in [7, 14, 15]. Inspired by the idea of exploiting edge caching to optimize communication performance in traditional communication, caching knowledge base at cloud, edge, and device to improve the performance of semantic communication becomes feasible. However, applying the knowledge base to semantic communication is still in its infancy, there are hardly any studies that give definitions of the knowledge base and state how to exploit knowledge in the semantic communication processes. Therefore, how to employ knowledge bases to assist semantic communications is still an open and challenging problem.

The most critical step to applying knowledge bases in a semantic communication system is by defining the content and usage scheme of knowledge bases. Knowledge graph has received extensive attention in recent years due to its strong performance in common sense and domain-specific knowledge applications [16], such as knowledge enhancement of knowledge graphs for text tasks [17] in natural language processing (NLP). Therefore, we exploit the knowledge graph as the main content of the knowledge base. With the assistance of a pre-cached knowledge base at the device, the receiver recovers the transmitted semantic information with an acceptable accuracy threshold. Consequently, the semantic communication system shall benefit from caching the knowledge graphs, alleviating the computing pressures of intelligence devices.

Existing semantic communication systems [7, 10–13] mainly focus on simulation implementation, some problems in applying semantic communication to actual wireless environments still remain to be solved. Since semantic vectors consist of float-type symbols, the existing binary transmission system cannot be directly used for semantic vector transmission. If semantic vectors are sent directly through a radio transmitter, there is no guarantee that these symbols can be orthogonal to each other, and the interference between the symbols will aggregate difficulty in the semantic recovery at the receiver. The transmission of semantic vectors is still an open problem to be solved in the semantic transmitter design.

Motivated by [18], we introduce orthogonal frequency division multiplexing (OFDM) into a semantic communication system and transform semantic vectors into orthogonal symbols to be directly transmitted over sub-carriers. Since neural networks have a certain level of tolerance to noise, the semantic vector can be recovered directly by a well-trained neural network. In this way, the semantic communication system will be more lightweight by bypassing some traditional signal processing techniques designed to ensure traditional transmissions, such as channel coding and modulation. By exploiting deep learning technologies to realize joint source-channel coding (JSCC), semantic communication can improve communication quality while reducing the overall complexity of the communication system.

Therefore, we propose a knowledge-enhanced semantic communication (KESC) system with OFDM transmissions. The main contributions of this paper can be summarized as follows:

(1) We propose a KESC system, where a cloud-edge-device collaboration architecture is considered to provide a knowledge base to synchronize common background knowledge of the transmitter and the receiver, where the knowledge base is a complete knowledge graph. The cloud server trains the knowledge graph into semantic vectors, and divides the knowledge graph and semantic vectors into sub-graph and sub-vectors. Furthermore, the edge server will divide the sub-graph and subvectors into multiple smaller sub-blocks, which can be cached in the devices to improve the performance of the system.

(2) Based on the proposed KESC system, we formulate a knowledge graph-based semantic information transmission scheme, where OFDM is exploited to simplify semantic information transmission. We design

a semantic pilot in each frame to assist semantic recovery, where traditional signal processing techniques are bypassed to support the semantic information transmission based on float-type symbols.

(3) Furthermore, a multi-encoder transformer based KESC for text transmission (KESC-T) model is proposed, which can simultaneously exploit the semantic information of both the source sentence and the knowledge graph, and perform knowledge enhancement on the received semantic vector. In addition, the model can further compress the vector dimension and thus reduce the number of transmitted bits.

(4) We define a new performance metric named knowledge-enhanced efficiency (KEE), which evaluates the performance of the knowledge-enhanced system and quantifies the gain in recovery accuracy brought by per unit of knowledge. Simulation results verify that our proposed KESC-T model outperforms the compared schemes, especially in low signal-to-noise ratio (SNR) and resource-constrained scenarios.

The rest of this paper is organized as follows. In Section 2, a brief review of related work is presented. The system model of our proposed KESC scheme is proposed in Section 3. Section 4 introduces the structure of our proposed KESC-T neural network. The relevant performance metrics are introduced in Section 5. The numerical results of KESC-T are presented in Section 6. Finally, Section 7 makes a conclusion of the paper.

2 Related work

2.1 Semantic communication systems

Shannon and Weaver pointed out in [19] that communication consists of three levels. The second level is the semantic level, aiming to accurately convey the meaning of the transmission content through communication. The authors of [20] investigated a model-theoretical approach for semantic data compression and reliable semantic communication. In order to cope with the transmission under multipath fading channels, a deep learning-based JSCC scheme for wireless image transmission with OFDM is proposed in [18]. Semantic communication systems for text transmission, speech transmission, and multi-modal data transmission are also designed in [7, 11, 12]. Moreover, a first understanding and then transmission framework with high semantic fidelity is exploited to audio transmission [13]. To reduce the size and computation of the deep learning-based semantic communication system (DeepSC) model, the methods of pruning and quantization are applied [10].

Furthermore, assuming that all devices share common knowledge, a novel semantic communication architecture was proposed in [14], which exploits federated edge intelligence to support resource-efficient semantic-aware networks. The authors in [15] believed the semantic base (Seb) should be defined first, and a dynamically-updated knowledge base should be deployed in the system, then propose an intelligent and efficient semantic communication (IE-SC) network architecture based on Seb to support 6G. Further, a model-driven semantic communication system is proposed in [21], which enables the communication system to evolve from traditional bit transmission to "model" transmission. The knowledge graph is proposed in [22] as transport symbols for semantic communication since the triple in the knowledge graph is a more general form of semantic organization and is inherently readable. In addition, the authors in [23] construct a knowledge graph to store the underlying relations between communication goals and semantic information, which enables computationally-efficient semantic extraction.

2.2 Knowledge enhancement based on knowledge graph

In order to realize a KESC system, it is necessary to determine the representation of knowledge. In recent years, knowledge graphs have attracted increasing attention. Knowledge graphs aim to model knowledge in a structured way and make efficient use of it. Generally speaking, a knowledge graph consists of numerous triples, each of which can be represented as (h, r, t), where h represents the head entity, t represents the tail entity, and r represents the relation between h and t. Entities in triples linked to each other form a network, which is known as the knowledge graph.

In the field of NLP, existing models are not enough to handle textual information with a lot of common sense. Since knowledge graphs contain a lot of common sense and domain-specific knowledge, knowledge graphs are proposed for NLP to enhance existing models with knowledge. With the assistance of knowledge graphs, an enhanced language representation model named ERNIE is formulated in [17] by modifying the traditional transformer encoder. The knowledge graph is injected into the pre-trained model bidirectional encoder representation from transformers (BERT) [24] to obtain a new model called K-BERT

in [25], which significantly outperforms BERT in domain-specific tasks. KG-BART was proposed in [26], in which the knowledge graph is used in encoder-decoder architecture, and knowledge enhancement is performed in the encoder and decoder respectively. In [27–31], knowledge graphs are used for machine translation tasks to enhance the semantic representation of the output text.

2.3 Multi-encoder transformer for multi-source input

Due to the complementary knowledge graph, KESC needs to solve the problem of multi-source input in neural networks. With the development of deep learning, multi-source sequence-to-sequence tasks have attracted widespread attention. Since the transformer performs well in sequence-to-sequence tasks, it is expected to be a good choice for multi-source sequence-to-sequence tasks. The authors in [32] analyzed the ability of multiple encoders, and believed that multiple encoders can improve the noise tolerance of the model and make the model more robust. Four combination strategies of serial, parallel, flat, and hierarchical for multi-encoder attention are proposed in [33], which are proven to be able to use multiple sources and outperform single source baselines. Moreover, the multi-encoder transformer employed in [34–36] is utilized to complete the automatic post-editing (APE) task, improve the text generation task and improve the performance of automatic speech recognition task, respectively.

3 Knowledge-based semantic communication system design

As shown in Figure 1, we propose a KESC system, where cloud-edge-device are collaborated to provide a knowledge base. The cloud server stores a huge knowledge base, which contains a knowledge graph $\mathcal{K}$ consisting of numerous triples. Meanwhile, the cloud server trains the triples in the knowledge base through its powerful computing ability and obtains the corresponding semantic embedding vectors $\mathcal{V}$. Due to the limited storage capability of the edge server and user device, the cloud server divides the knowledge graph $\mathcal{K}$ and the corresponding semantic vectors $\mathcal{V}$ into multiple sub-blocks and obtains $\mathcal{K} = [\mathcal{K}_1, \mathcal{K}_2, \ldots, \mathcal{K}_n]$, $\mathcal{V} = [\mathcal{V}_1, \mathcal{V}_2, \ldots, \mathcal{V}_n]$, where $\mathcal{K}_l$ represents the l-th sub-graph, $\mathcal{V}_l$ represents the l-th sub-vectors, n represents the number of the divided blocks.

The user devices record each request to the device knowledge base, and periodically feedback on the records to the edge server. Then the edge server periodically sends update requests to the cloud server based on users' request frequency. At this time, the cloud server will distribute the divided knowledge graph and corresponding semantic vector to each edge server within range, generating a unique version number for the divided knowledge base, which plays a role in the subsequent semantic decoding. Furthermore, the edge server can divide the sub-graph $\mathcal{K}_l$ and the corresponding sub-vectors $\mathcal{V}_l$ into multiple smaller sub-blocks, which will be distributed to the user device according to the update requests of user devices. Meanwhile, the cloud knowledge base is updated as the non-private data sent by users, to improve the service quality of semantic communication. For the specific sub-block division strategy, under the premise of limited cache space, the server will select the sub-graphs or triples with the highest value according to the requests of nodes to maximize the value of sub-blocks.

Based on the aforementioned knowledge base located in the device, edge server, and cloud, we design a KESC system with the OFDM transmission scheme, which is shown in Figure 2. The main process of KESC schemes includes semantic encoding, semantic pilot generation, time-domain symbols generation, transmission over wireless channels, and semantic decoding.

3.1 Effective semantic encoder: joint design of semantic encoder and semantic compression

Based on the proposed scheme in Figure 2, the user device can send information with multiple modalities. In this paper, we focus on text transmission over wireless channels. The transmitter first performs entity recognition on the text data $\boldsymbol{s}$ to obtain the entity list $\boldsymbol{E} = [e_1, e_2, \ldots, e_n]$, where e_l represents the l-th entity in the $\boldsymbol{s}$, n represents the number of the entities. The entity list is used to search for the triples in the knowledge base and generate the semantic pilot. The neural network model is used to convert the text data $\boldsymbol{s}$ into semantic vector $\boldsymbol{v_s}$ and performs a dimension compression process to obtain the semantic symbols $\boldsymbol{X_s}$. The overall encoding process can be expressed as

$$\boldsymbol{v_s} = S_{\text{enc}}(\boldsymbol{s}), \quad \boldsymbol{X_s} = C_{\text{enc}}(\boldsymbol{v_s}), \tag{1}$$

where S_{enc} represents the semantic encoder model and C_{enc} represents the semantic compression model.

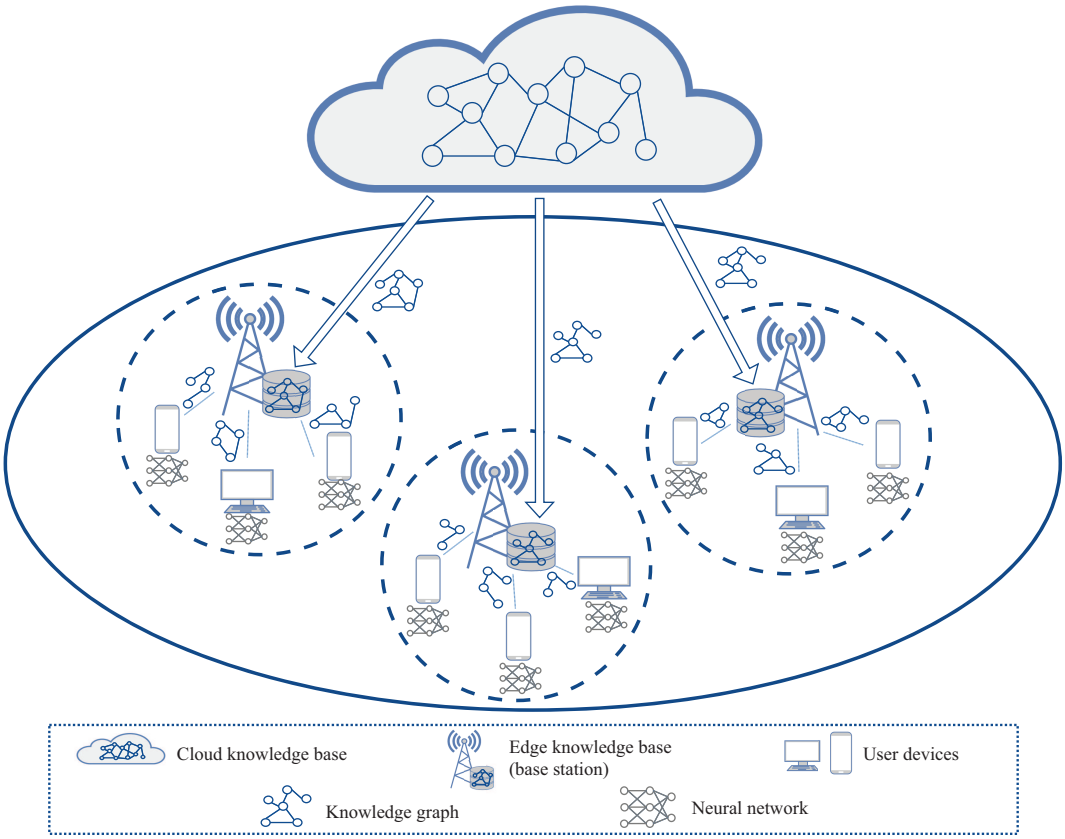

Figure 1 (Color online) The proposed KESC framework.

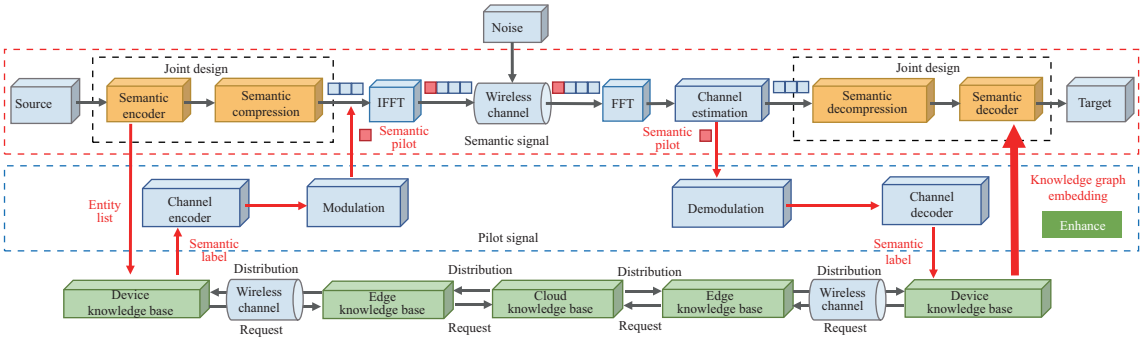

Figure 2 (Color online) The proposed KESC scheme with OFDM.

3.2 Knowledge-based semantic transmitter design

Based on the entity list generated by the transmitter, the device knowledge base queries the corresponding triples $\boldsymbol{T} = [t_1, t_2, \ldots, t_n]$, where t_l represents the l-th triple and n represents the number of the triples. However, the device knowledge base may not query all triples corresponding to the entities due to the part of the knowledge graph cached. Therefore, we define a semantic pilot $\boldsymbol{X}_{\rm sp}$, as the combination of the current knowledge base version number and the semantic label of the triple. The role of the semantic pilot is to ensure that the receiver can obtain the correct triples for knowledge enhancement with the least transmission bits. The semantic pilot $\boldsymbol{X}_{\rm sp}$ is generated from the label of triples by Algorithm 1 if the device successfully finds the triple. If the device failed to find the triple, it will set the semantic pilot to zero for transmission. At the same time, the device records the request frequency of each entity, so as to request the corresponding knowledge graph from the edge server base on the recorded frequency. As for the semantic labels, their specific content is closely related to the storage structure of the knowledge graphs. The content can be triple IDs for relational database-based storage while node IDs and relation

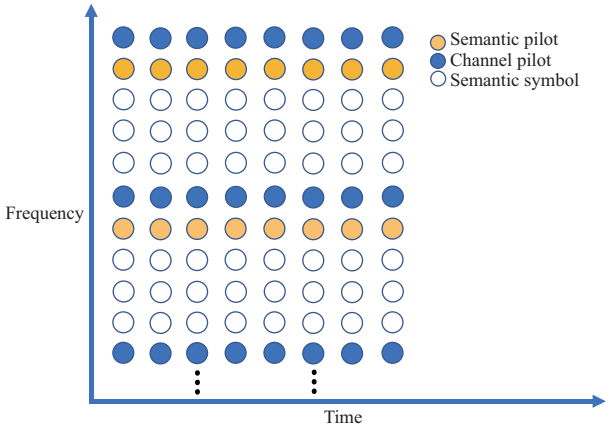

Figure 3 (Color online) The proposed semantic pilot arrangement.

Algorithm 1 Semantic pilot generation algorithm

Input: Entity list $\boldsymbol{E} = [e_1, e_2, \ldots, e_n]$.
Output: Semantic pilot list $\boldsymbol{P} = [\boldsymbol{X}_{\mathrm{sp}_1}, \boldsymbol{X}_{\mathrm{sp}_2}, \ldots, \boldsymbol{X}_{\mathrm{sp}_n}]$.

```
Initialize pilot list P = [ ];
for All triples (h, r, t) in knowledge base do
    if h in E and t in E then
        Encode the semantic label l of the triple to x_sp;
        Modulate x_sp to frequency domain to be x_sp;
        Add the semantic pilot x_sp to the list P;
    end if
end for
```

edge IDs for native graph-based storage. Regardless of the storage structure, semantic labels will be generated directly when new triples are inserted.

However, since the transmitted data is a vector of float-type symbols, it will cause a considerable overhead when transmitted in a traditional communication system. If the float-type symbols are converted into binary-type before transmission, it will not only affect the symbol accuracy but also result in a large growth in the processing delay. Therefore, inspired by [18], we combine float-type symbols in pairs and convert them to complex symbols, which can be seen as frequency-domain symbols in traditional communications. Furthermore, we use OFDM to directly perform IFFT (inverse fast Fourier transform) and transform complex symbols into time-domain symbols, where the transmission vectors are orthogonal to each other.

To ensure successful decoding for each corresponding sentence at the receiver, learning from the pattern of comb-type pilots, we specify the symbol length of each sentence and insert semantic pilots at the beginning of each sentence. A specific pilot example is shown in Figure 3. To achieve physical denoising before decoding, both semantic pilot and traditional pilot symbols used for channel estimation are added before the semantic symbols. It can be seen as the process of combining semantic symbols $\boldsymbol{X_s}$ with semantic pilot $\boldsymbol{X}_{\mathrm{sp}}$ and channel pilot $\boldsymbol{X}_{\mathrm{cp}}$, expressed as

$$\boldsymbol{X} = \mathrm{Concat}(\boldsymbol{X}_{\mathrm{cp}}, \boldsymbol{X}_{\mathrm{sp}}, \boldsymbol{X_s}). \tag{2}$$

All semantic symbols will be transmitted in float-type for direct communication. Traditional channel pilot symbols will be sent via the conventional binary bit communication method. After a series of processes such as channel coding and modulation, the traditional pilot symbols will be transformed to float-type, superimposed, and transmitted together with the semantic symbols and the semantic pilot.

Finally, the serial data stream including semantic vector, semantic pilot, and channel pilot will be transformed into time-domain symbols $\boldsymbol{x}$ by IFFT transformation and is written as

$$\boldsymbol{x} = \mathrm{IFFT}(\boldsymbol{X}). \tag{3}$$

After that, the time-domain symbols $\boldsymbol{x}$ will be transmitted through a wireless channel, and the signals received at the receiver are expressed as

$$\boldsymbol{y} = h * \boldsymbol{x} + n, \tag{4}$$

where * denotes convolution operation, h is the channel gain of the fading channel, and n is the noise of additive white Gaussian noise (AWGN) channel.

3.3 Effective semantic decoder: joint design of semantic decompression and semantic decoding

At the receiver, FFT is performed on the received OFDM symbol stream after synchronization

$$\boldsymbol{Y} = \mathrm{FFT}(\boldsymbol{y}). \tag{5}$$

After channel estimation and channel compensation, the semantic pilot $\boldsymbol{Y}_{\mathrm{sp}}$ is taken out and decoded to obtain the corresponding knowledge embedding. The specific knowledge query process is shown in Algorithm 2. If the device finds that the current knowledge base version does not match the version in the pilot, it will directly input the received semantic symbols $\boldsymbol{Y}_{\boldsymbol{s}}$ into the decoder for recovery. Otherwise, the semantic label in the pilot will be used to query the knowledge embedding $\boldsymbol{v}_k$ corresponding to each sentence. Meanwhile, the knowledge embedding $\boldsymbol{v}_k$ and the received semantic symbols $\boldsymbol{Y}_{\boldsymbol{s}}$ will be decoded together for semantic recovery. The recovery process includes semantic decompression and semantic decoding, which can be written as

$$\hat{\boldsymbol{v}}_{\boldsymbol{s}} = C_{\mathrm{dec}}(\boldsymbol{Y}_{\boldsymbol{s}}), \quad \hat{\boldsymbol{s}} = S_{\mathrm{dec}}(\hat{\boldsymbol{v}}_{\boldsymbol{s}}, \boldsymbol{v}_k), \tag{6}$$

where C_{dec}, S_{dec}, $\hat{\boldsymbol{s}}$ are the semantic decompression model, semantic decoder model, and final recovered symbol, respectively.

Algorithm 2 Knowledge query algorithm

Input: Semantic pilot list $\boldsymbol{P} = [\boldsymbol{X}_{\mathrm{sp}_1}, \boldsymbol{X}_{\mathrm{sp}_2}, \ldots, \boldsymbol{X}_{\mathrm{sp}_n}]$.
Output: Knowledge embedding list $\boldsymbol{V} = [\boldsymbol{v}_{k_1}, \boldsymbol{v}_{k_2}, \ldots, \boldsymbol{v}_{k_n}]$.

```
Initialize embedding list V = [ ];
for all pilot X_sp in P do
    Demodulate X_sp to time domain to be X_sp;
    Decode the pilot X_sp to be the semantic label l;
    if the version of knowledge base in l match then
        if the triple of l is found in the knowledge base then
            Embedding the triple of the label l as v_k;
            Add the embedding v_k to the list V;
        else
            Add an all-zero-valued vector v_k to the list V;
        end if
    end if
end for
```

4 Multi-encoder transformer based neural network model for KESC

In order to realize the KESC system using the device knowledge base, we propose a neural network model based on a multi-encoder transformer for KESC-T, which is shown in Figure 4.

4.1 Semantic encoder module

Data preprocessing is a key part of semantic communication. Before entering the semantic encoder, a large amount of text data generated by the user device will perform a series of preprocessing operations including data cleaning, sentence segmentation, and word tokenization. Each word in the sentence will be mapped to a corresponding number according to the vocabulary, the processed sentence can be represented as $\boldsymbol{s} = [w_1, w_2, \ldots, w_L]$, where w_l is the l-th word in the sentence, L represents the length of the sentence.

The semantic encoder in our proposed KESC-T model adopts the classic transformer encoder structure, which includes a token embedding layer, position embedding layer, and encoder-blocks. The processed sentence $\boldsymbol{s}$ will be fed into the token embedding layer to generate token embedding $\boldsymbol{v}_e$. Each word in the input sentence $\boldsymbol{s}$ will be mapped to a vector of dimension K in the latent space, and the entire sentence can be written as a vector matrix $\boldsymbol{v}_e \in \mathbb{R}^{L\times K}$. Furthermore, to ensure that the encoder can identify the order information of each word in the sentence, the sentence will be fed into the position embedding layer

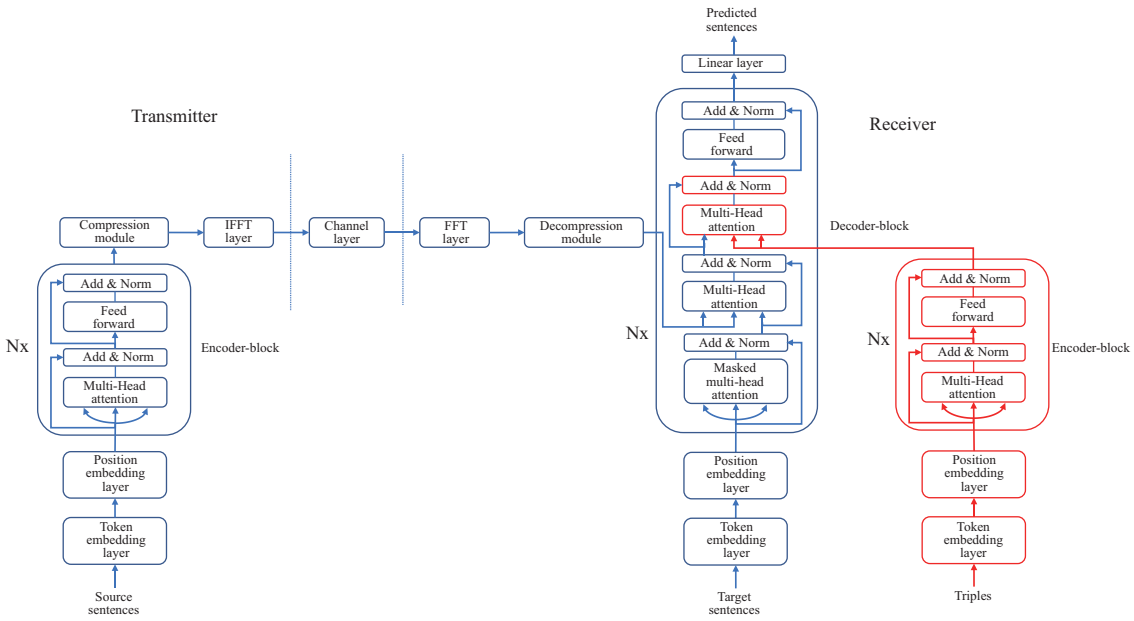

Figure 4 (Color online) The proposed neural network structure for the semantic communication system.

to encode the position of each word in the sentence, generating vector $\boldsymbol{v}_p$. The results of word embedding $\boldsymbol{v}_e$ and position embedding $\boldsymbol{v}_p$ will be summed as the input of the encoder-blocks.

The encoder consists of multiple encoder-blocks, which can effectively extract semantic associations of words in sentences. Each layer contains a feed-forward layer and a self-attention layer. The self-attention process can be expressed as

$$\text{Attention}(\boldsymbol{Q_s}, \boldsymbol{K_s}, \boldsymbol{V_s}) = \text{softmax}\left(\frac{\boldsymbol{Q_s}\boldsymbol{K_s}^{\text{T}}}{\sqrt{d}}\right)\boldsymbol{V_s}, \tag{7}$$

where $\boldsymbol{Q_s}$, $\boldsymbol{K_s}$, and $\boldsymbol{V_s}$ represent the queries, keys, and values set, respectively, T stands for the matrix transpose. $\boldsymbol{Q_s}$, $\boldsymbol{K_s}$, $\boldsymbol{V_s}$ are all generated from the source sentence $\boldsymbol{s}$, and d represents the dimension of the queries and keys. The queries $\boldsymbol{Q_s}$, keys $\boldsymbol{K_s}$, and values $\boldsymbol{V_s}$ are all the specific variables to realize the attention in transformer.

Furthermore, residual connections and layer normalization are also applied in each of the two sub-layers to make the network more robust. As a high-dimensional representation of the entire sentence, the output of the final sub-layer will be the semantic encoder output $\boldsymbol{v_s} \in \mathbb{R}^{L\times K}$.

4.2 Compression and quantization module

To reduce the scale of transmission, we adopt dimension compression technology and data quantization technology, which are designed as follows.

Dimension compression. We use a classic autoencoder structure based on deep neural networks (DNN) as the compression network. The high-dimensional vector matrix will be sent to the network for dimension compression. The neurons in the encoder part of the autoencoder are decreased layer by layer, so the semantics in each dimension can be easily extracted for integration and compression to form a low-dimensional vector $\boldsymbol{X_s} \in \mathbb{R}^{L\times D}$, where D represents the number of neurons in the compression output layer. While the neurons in the decoder part of the autoencoder are increased layer by layer, the compressed vectors can be easily restored to the vectors $\hat{\boldsymbol{v}}_{\boldsymbol{s}} \in \mathbb{R}^{L\times K}$ with original dimension.

Data quantization. Since the output of the vector in the neural network is all 32-bit float-type, there is a large redundancy in representing semantic vectors. We quantize the output vector and convert the data type to 8-bit int-type, which reduces the transmission bits by a factor of four. The quantization and dequantization functions can be formulated as

$$\boldsymbol{X}_{\text{quantize}} = \text{round}((2^M - 1)(\text{Sigmoid}(\boldsymbol{X}))), \tag{8}$$

$$\boldsymbol{X}_{\text{dequantize}} = \frac{\boldsymbol{X}_{\text{quantize}}}{2^M - 1}, \tag{9}$$

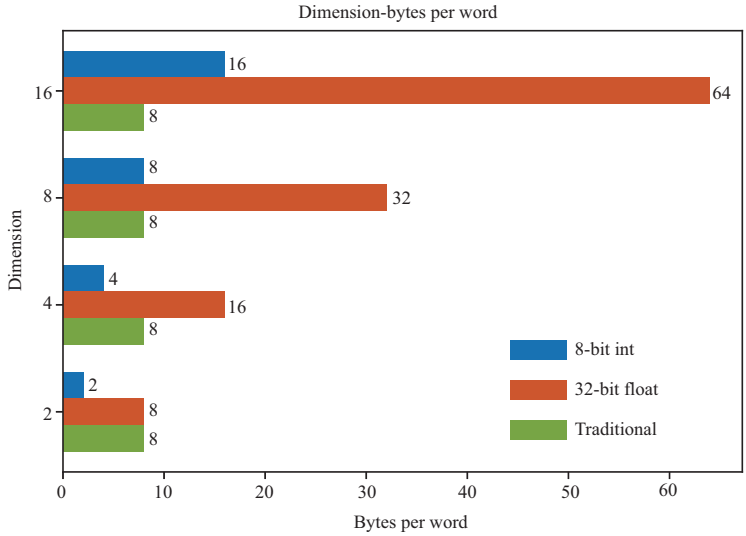

Figure 5 (Color online) The transmitted bytes versus different dimensions.

where M represents the number of bits after quantization. As for the quantization function, We first use the Sigmoid function to limit the range of the output vector between 0 and 1, which simplifies the quantization and dequantization operations. Theoretically, it can be proved that when the condition of $2^{M \cdot D} > N$ is satisfied, all words in the vocabulary can be represented by M bits, where N represents the size of the vocabulary. However, due to the noise and the limitation of model recognition accuracy in practical applications, a certain space is usually reserved for the tolerance of recognition error. It can be inferred that with a higher difference between $2^{M \cdot D}$ and N, the noise tolerance of the neural network model will be larger and the final classification will be more accurate.

As we all know, if each English letter is encoded in 8-bit Unicode transformation format (UTF-8), it will cost 1 byte of storage, while in the case of Unicode encoding, 2 bytes will be needed. Therefore, we use UTF-8 as the traditional encoding method. As shown in Figure 5, we compare the transmitted bytes under three communication modes: 8-bit int, 32-bit float, and traditional. Assuming that the average length of English words in the dataset is 8, if a sentence is transmitted by the traditional method, it will increase the redundant information in channel coding. Without considering the number of auxiliary symbols, the transmission cost of each word is at least 8 bytes. When transmitted according to semantic communication, each word can be represented by a D-dimensional vector. In the case that the 32-bit float type is directly used for transmission without quantization, more communication resources will be consumed than that in traditional occasions. If the semantic vector is quantized into the 8-bit and the final compressed dimension is less than 8, the transmission amount will be smaller than that of the traditional method without considering the number of auxiliary symbols.

4.3 Signal processing module: IFFT layer, channel layer, and FFT layer

In order to verify the feasibility of our proposed semantic communication systems, we apply signal processing layers containing an IFFT layer, a channel layer, and an FFT layer to the model. Furthermore, to achieve an actual signal processing procedure, the semantic vector output by the semantic encoder will go through the following processes in sequence.

(1) Transmitted symbols construction. As dimension compression and data quantization are complete, the vectors will be combined in pairs to form frequency-domain complex symbols, and the channel estimation pilot and semantic pilot are inserted into each sentence at the corresponding position.

(2) IFFT transformation. To transform the transmitted symbols into the time domain, the symbols will be further conveyed to an IFFT layer for transformation.

(3) Channel simulation. A noise channel layer is added to simulate different channel characteristics. Besides, the symbols will be sent to the noise channel layer to simulate actual channel transmission.

(4) FFT transformation. An FFT layer is subsequently applied to the received symbols after passing through the channel layer. Since both IFFT/FFT and channel layers can be regarded as differentiable but non-trainable neural network layers, the parameters can be back-propagated during model training.

(5) Channel estimation and compensation. Once the FFT transform is finished, the channel estimation is performed. After that, the channel compensation will be applied to the overall symbols.

(6) Semantic vector reconstruction. Following channel compensation, the semantic pilot will be extracted from the received complex symbols. Moreover, the semantic symbols will be re-transformed and dequantized to become a real vector matrix $\boldsymbol{Y_s} \in \mathbb{R}^{L\times D}$. Then, the vector matrix will be fed into the decoder part of the autoencoder, and the symbols can be restored to a vector matrix $\hat{\boldsymbol{v}}_{\boldsymbol{s}} \in \mathbb{R}^{L\times K}$.

4.4 Knowledge encoder module

Compared with the encoder-decoder structure of the classic transformer, the knowledge encoder layer proposed in our KESC-T model ensures that the additional knowledge used in the semantic decoder can be extracted efficiently. Therefore, we adopt the same structure as the semantic encoder to the knowledge encoder layer. Since the function of the triple embedding is to provide semantic information to assist the recovery of the receiver, instead of adopting the conventional method of knowledge graph embedding to embed the entire entity and relationship, we exploit token embedding to embed each token in the triple consistent with sentence input.

When the receiver gets the semantic pilot $\boldsymbol{X}_{\rm sp}$, the corresponding triples will be queried in the knowledge base according to the semantic pilot $\boldsymbol{X}_{\rm sp}$. The obtained triples will be embedded and input into the knowledge encoder with token embedding and position embedding. Then, the triple will be transformed into a vector matrix summed by $\boldsymbol{v}_e \in \mathbb{R}^{T\times K}$ and $\boldsymbol{v}_p \in \mathbb{R}^{T\times K}$ and input into the self-attention layer, where T represents the length of the triple. The self-attention process can be expressed as Attention$(\boldsymbol{Q}_k, \boldsymbol{K}_k, \boldsymbol{V}_k)$, where $\boldsymbol{Q}_k$, $\boldsymbol{K}_k$, and $\boldsymbol{V}_k$ are all generated from the triple and represent the queries, keys, and values set respectively, d represents the dimension of the queries and keys. The output of the final sub-layer is the knowledge encoder output $\boldsymbol{v}_k \in \mathbb{R}^{T\times K}$ as a high-dimensional representation of the entire triple.

4.5 Semantic decoder module

In order to understand the semantic information of the source sentence and the triples at the same time, motivated by [33], we improve the structure of the transformer decoder in our KESC-T model. Compared with the decoder of the classic transformer structure, we add a knowledge-target attention layer, which focuses on semantic extraction of the vectors input by the knowledge encoder. This layer is serially connected with the source-target attention layer in the classic transformer decoder and simultaneously extracts the semantic information of sentences and triples.

For the two different modes of training and inference, the decoder has two input methods. In training mode, the entire preprocessed sentence will be fed into the decoder with a mask to ensure that the attention can only get the information before the current predicted word. While in inference mode, a predefined ⟨START⟩ mark will be fed into the network first, which marks the beginning of the semantic recovery of the decoder. After that, the words will be predicted by auto-regression, and the predicted word in each round will be added to the previous input and put into the decoder together until the decoder generates the predefined ⟨END⟩ mark.

The semantic decoder also consists of three kinds of layers: token embedding layer, position embedding layer, and decoder-block. The token embedding layer and the position embedding layer are the same as the encoder. Similar to the encoder, the target sentence will be sent to the word embedding layer and the position embedding layer in turn to get the processed data $\boldsymbol{v}_e \in \mathbb{R}^{L\times K}$ and $\boldsymbol{v}_p \in \mathbb{R}^{L\times K}$. The output vector of the two layers will be summed and input to the decoder-block.

Different from the encoder, the decoder-block consists of a self-attention layer, a source-target attention layer, a knowledge-target attention layer, and a feed-forward layer. The vector will firstly go through the self-attention layer, expressed as Attention$(\boldsymbol{Q}_t, \boldsymbol{K}_t, \boldsymbol{V}_t)$, where $\boldsymbol{Q}_t$, $\boldsymbol{K}_t$, $\boldsymbol{V}_t$ are all generated from the target sentence and represent the queries, keys, and values set respectively.

The output of the self-attention layer and semantic encoder will be fed into the source-target attention layer simultaneously. The source-target attention process can be expressed as Attention$(\boldsymbol{Q}_t, \boldsymbol{K_s}, \boldsymbol{V_s})$, where $\boldsymbol{Q}_t$ represents the queries set generated from the target sentence, $\boldsymbol{K_s}$ and $\boldsymbol{V_s}$ represent the keys and values set generated from the semantic encoder outputs, respectively.

After all the processes above, the output of the source-target attention layer and knowledge encoder will be fed into the knowledge-target attention layer together. The knowledge-target attention process

Table 1 The complexity of KESC-T

Module	Complexity per layer	Sequential operations	Maximum path length
Attention layer	$O(n^2 \cdot d)$	$O(1)$	$O(1)$
Feed-forward layer	$O(n \cdot d^2)$		

can be expressed as Attention($\boldsymbol{Q}_t, \boldsymbol{K}_k, \boldsymbol{V}_k$), where $\boldsymbol{Q}_t$ represents the queries set generated from the target sentence, $\boldsymbol{K}_k$ and $\boldsymbol{V}_k$ are the keys and values set generated from the knowledge encoder outputs.

The vector generated from the target sentence is always used as a query. Query operations will be performed on the vectors based on different attention requirements, and semantic information will be extracted respectively and input into the linear layer for semantic integration. The number of the corresponding words in the vocabulary will also be predicted. The label of each word in the output sentence will be remapped according to the vocabulary to generate the final sentence.

4.6 Complexity analysis

To illustrate the computation time requirements of the KESC-T, we analyze the two core components of the KESC-T base on the method in [37], the analysis results are shown in Table 1. It can be observed that the computational complexity of the attention layer is $O(n^2 \cdot d)$, while that of the feed-forward layer is $O(n \cdot d^2)$, where n represents the sentence length and d represents the hidden dimension. This result shows that the feed-forward layer occupies most of the computing time when the sentence length is short, while the attention layer occupies most of the computing time when the sentence length is long. Although these two layers are both the main components of the whole system, the attention layer has a larger influence on the performance of our proposed KESC-T model, which verifies that our proposed KESC-T model has better performance in dealing with short sentences.

Since the sequence operation of the attention layer can be completed in parallel, the sequence operation complexity of KESC-T can be evaluated as $O(1)$. Furthermore, we use the maximum path length defined in [38] to measure the degree of information loss when nodes with long-distance dependencies transmit information. The maximum path length is the longest distance required by two nodes in the neural network to transmit information. The interaction between two nodes will be more difficult and the information loss becomes more severe with the path length increasing. Therefore, the complexity of searching the maximum path length in the attention layer is $O(1)$ because nodes in the attention layer can be directly connected with each other.

5 Performance metrics

Since semantic communication only requires the receiver to recover the semantic information sent by the transmitter rather than requiring strict recovery of each bit, the metrics used to evaluate traditional communication systems are not effective. In order to evaluate the recovery degree of semantic information, we adopt three existing performance metrics in NLP and propose a new performance metric to measure our proposed KESC system.

(1) BLEU [8]. BLEU evaluates the difference in two sentences by comparing the n-grams of the predicted sentence with the reference sentence and then calculates the degree of coincidence, where n-grams represent the size of a word group. The accuracy of semantic recovery depends on the difference between two sentences. As the degree of coincidence increases, the difference between two sentences gets smaller, while the accuracy of semantic recovery rises higher. Usually, 1-gram is used to compare the accuracy of individual words, and 2-grams, 3-grams, and 4-grams are used to measure the fluency of sentences. The BLEU formulated in the log domain is expressed as

$$\log(\text{BLEU}) = \min\left(1 - \frac{l_{\hat{s}}}{l_s}, 0\right) + \sum_{n=1}^{N} u_n \log(p_n), \tag{10}$$

where $l_{\boldsymbol{s}}$ is the length of the reference sentence $\boldsymbol{s}$, $l_{\hat{\boldsymbol{s}}}$ represents the length of the predicted sentence $\hat{\boldsymbol{s}}$, u_n denotes the weights of n-grams, p_n denotes the n-grams score and is written as

$$p_n = \frac{\sum_k \min(C_k(\hat{\boldsymbol{s}}), C_k(\boldsymbol{s}))}{\sum_k \min(C_k(\hat{\boldsymbol{s}}))}, \tag{11}$$

where $C_k(\cdot)$ is the frequency count function for the k-th elements in n-th grams.

(2) Sentence similarity [7]. Sentence similarity evaluates the differences between two sentences from sentence level rather than word level. Therefore, the problem of words with variable meanings in different contexts is solved. Sentence similarity is defined as

$$\text{Similarity}(\hat{\boldsymbol{s}}, \boldsymbol{s}) = \frac{B_\phi(\boldsymbol{s}) \cdot B_\phi(\hat{\boldsymbol{s}})}{\|B_\phi(\boldsymbol{s})\| \, \|B_\phi(\hat{\boldsymbol{s}})\|}, \tag{12}$$

where B_ϕ represents a pre-trained model used for extracting semantic information. It can output a vector representation for the whole sentence, and calculate the similarity of two sentences by cosine similarity.

(3) WER [9]. WER is defined as the normalized Levenshtein distance, measuring the similarity between two strings by calculating the minimum number of operations required to convert from one string to another. WER is expressed as

$$\text{WER} = \frac{S + D + I}{N} = \frac{S + D + I}{S + D + C}, \tag{13}$$

where S, D, I, C, N represent the number of substitutions, the number of substitutions deletions, the number of substitutions insertions, the number of substitutions correct words, and the number of substitutions words in the reference, respectively.

(4) KEE. Since our proposed KESC system is knowledge enhanced, in order to measure the efficiency of knowledge utilization in our proposed KESC system, we define KEE to quantify the improved performance of recovery accuracy (such as BLEU) by exploiting per unit of knowledge. In this paper, we consider BLEU as the recovery accuracy and define KEE as

$$\text{KEE} = \tanh\left(\frac{\text{BLEU}(\alpha, \beta) - \text{BLEU}(\alpha, \gamma)}{\text{BLEU}(\alpha, \tau)}\right), \tag{14}$$

where α, β, γ, and τ denote the reference sentence, the predicted sentence recovered by the knowledge-enhanced system, the predicted sentence recovered by the normal system, and the triples used for knowledge enhancement, respectively.

We exploit tanh function to normalize the value of KEE into $[-1, 1]$, where $\tanh(x) = \frac{\sinh(x)}{\cosh(x)} = \frac{\mathrm{e}^x - \mathrm{e}^{-x}}{\mathrm{e}^x + \mathrm{e}^{-x}}$. When KEE is positive, the knowledge benefits recovery, and the trained model performs better with KEE closing to 1. When KEE is negative, the knowledge is treated as noise for recovery and has a negative influence on semantic recovery. Therefore, the trained model performs worse as KEE approaches -1.

Moreover, KEE is also applicable to other knowledge-enhanced systems. If KEE is used in an image semantic communication system, it only replaces BLEU with the corresponding evaluation metric. In our KESC-T model, KEE measures the average contribution of each token in the triples to improve semantic recovery. Namely, the amount of containing information increases with the number of tokens in the triple, while the contribution decreases under the premise of the same semantic gain.

6 Simulation results

6.1 Dataset

To verify the performance of the proposed KESC system, we adopted ReVerb45K [39] as the dataset. ReVerb45K is a canonicalization dataset based on Open KB. For each triple in ReVerb Open KB [40], the corresponding source text is extracted from the Clueweb09 corpus [41], obtaining a high-quality dataset with both the source text and corresponding triple. Therefore, the dataset of ReVerb45K is well suited for our proposed KESC system. We extract the source text and triples in ReVerb45K respectively, and discard the source text with a length of more than 30 words and corresponding triples. Statistics show that the remaining data contains 89055 pieces of sentences, and the vocabulary contains 39834 words. We generate a training set and test set according to the ratio of 8 : 2. The training set contains 71244 pieces of sentences, and the test set contains 17811 pieces of sentences.

6.2 Simulation settings

The experimental environment is Ubuntu 20.04, the GPU is Nvidia Tesla V100 32G, and Pytorch is applied as the deep learning framework. In order to balance the improvement of recovery accuracy and

Table 2 The setting of semantic network

Part	Layer name	Units	Activation
Semantic encoder	4 × transformer encoder-blocks	300 (10 heads)	Linear
Semantic compression	Linear layer	300	Elu
	Linear layer	16	
Semantic decompression	Linear layer	16	Elu
	Linear layer	300	
Knowledge encoder	4 × transformer encoder-blocks	300 (10 heads)	Linear
Semantic decoder	4 × improved transformer decoder-blocks	300 (10 heads)	Linear

the consumption of computing resources, we adopt 4 transformer encoder-blocks connected in series as the semantic encoder and knowledge encoder. In addition, we use 4 improved transformer decoder-blocks connected in series as the semantic decoder. For the embedding layer of the model, since the vocabulary at the transmitter and receiver is unified, we share the layer parameters with the embedding layers of the semantic encoder, knowledge encoder, and semantic decoder. Besides, we employ the pre-trained word embedding GloVe [42] instead of the random word embedding to provide better semantic features. Specifically, we choose the 300-dimension version trained on 42 billion tokens of Common Crawl data, since the vocabulary of this version has the highest overlap with the vocabulary of ReVerb45K. In the semantic compression and semantic decompression layer, we apply the DNN-based autoencoder structure to compress and restore the semantic vector matrices. In order to make the neural network more robust, we randomly add 0 to 6 dB Gaussian white noise to the network during training. For best model performance, the optimal parameters of the whole model are shown in Table 2.

We choose the cross-entropy function as the loss function to evaluate the difference between two probability distributions. The cross-entropy function can be formulated as

$$L_{\mathrm{CE}}(\boldsymbol{s}, \hat{\boldsymbol{s}}) = -\sum_{l=1} q(w_l)\log(p(w_l)) + (1-q(w_l))\log(1-p(w_l)), \tag{15}$$

where $q(w_l)$ is the real probability of the l-th word w_l appears in the estimated sentence $\boldsymbol{s}$, and $p(w_l)$ is the predicted probability of the l-th word w_l in the sentence $\hat{\boldsymbol{s}}$.

Furthermore, we employ the Adam optimizer to improve the optimization performance with an adaptive learning rate. We found that with a smaller batch size, the final loss value of training will converge to a higher position. With the limitation of GPU performance, we choose 512 as the size of each batch to meet the accuracy requirements of training. To evaluate our proposed KESC-T model from different perspectives, the performance of BLEU, sentence similarity, WER, and KEE are evaluated.

6.3 Results analysis

In order to verify that the addition of a knowledge base is beneficial to the semantic communication system, we adopt the DeepSC model [7] as the typically compared scheme. Moreover, we adopted a traditional transmission method as the baseline scheme, which uses 5-bit for fixed-length source coding and convolutional codes for channel coding, and 64-QAM is used as the modulation method.

Figure 6 shows the relationship between the BLEU score and the SNR over AWGN, Rayleigh, and Rician channels. It can be seen that the BLEU in our proposed KESC-T model is higher than that of DeepSC under all channel conditions, which verifies that the additional knowledge is beneficial to semantic recovery. The proposed KESC-T model outperforms the traditional method in low-SNR scenarios, while the traditional method performs better in high-SNR scenarios. As a traditional signal processing technique, channel coding has a certain error correction capability, especially in high-SNR scenarios. Moreover, the error correction capability of the channel coding will not be affected by the change in the transmission sentence. However, the semantic communication system is limited by the effect of neural network training, and the effect of neural network training is closely related to the difficulty of the training dataset. Therefore, the performance of semantic communication is not as good as that of traditional communication in high-SNR scenarios. This situation will be improved as the dataset is cleaner or the network structure is more powerful. Since 0 to 6 dB random Gaussian noise is added to the transmitted information during training, the semantic recovery accuracy of the KESC model can maintain stability under various SNR scenarios. As for the sharp rise in performance of "5-bit+cov", it may be due to the limited performance of convolutional codes. As one of the simplest channel codes, convolutional codes are

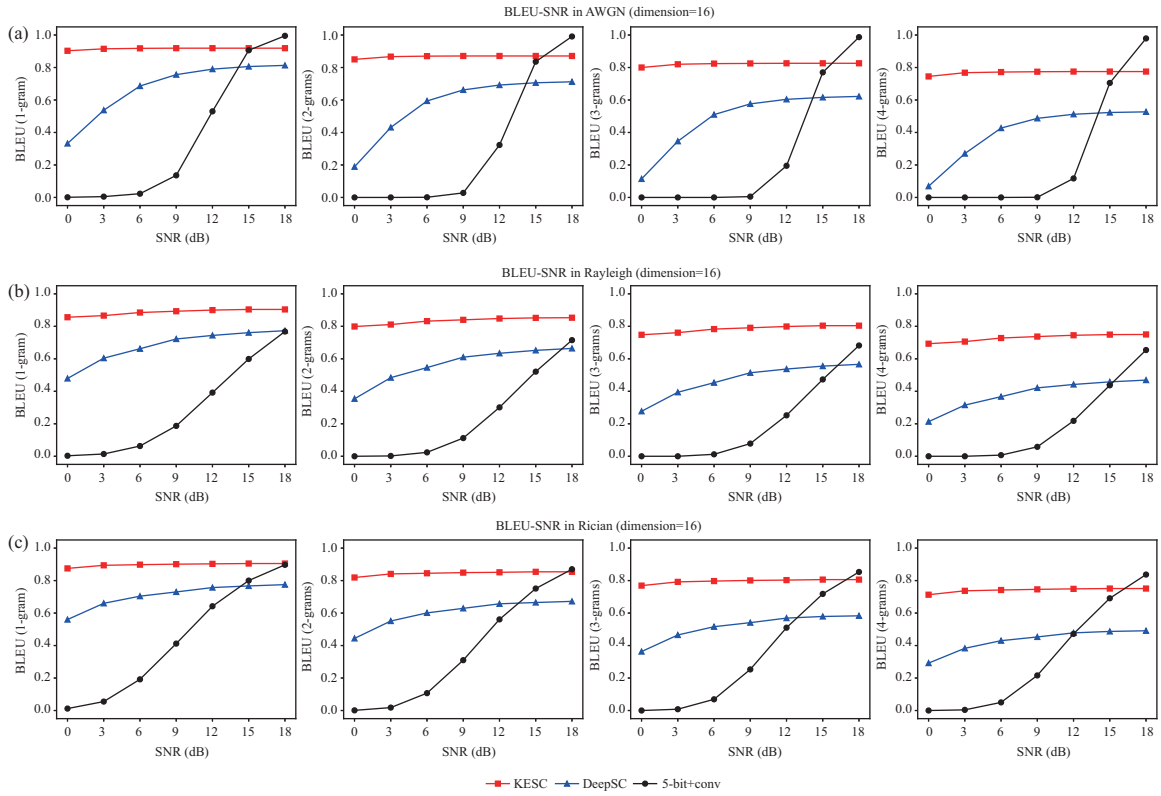

Figure 6 (Color online) The BLEU score versus SNR in (a) AWGN, (b) Rayleigh, and (c) Rician channels.

Table 3 The sample sentence in AWGN channel when SNR is 12 dB

Conditions	Contents
Transmitted sentence	south africa is the richest country in africa and could be one of the richest countries in the world
Relevant triple	[“south africa”, “be the richest country in”, “africa”]
KESC-T	south africa is the richest country in africa and could be one of the richest countries in the world
DeepSC	south africa is the developer country in africa and could be one of the section countries in the world
5-bit+cov	south africa ie the d.cjxst countrybqn africa and could be one ofg;pe rhwhest countries in the world

not very capable of correcting errors in low-SNR scenarios. If a more advanced channel coding technique, such as LDPC codes, is used, it is believed that the performance of such abrupt increases or decreases will be mitigated.

Table 3 shows an example of sentence recovery when SNR is 12 dB. Furthermore, we find that the BLEU of our proposed scheme has never been able to break through the level achieved in [7] by DeepSC, which is caused by the adopted dataset. The accuracy of the neural network is closely related to the amount of training data. Under the premise that the amount of data is roughly the same, word recovery gets harder as the vocabulary gets larger. In the Europarl dataset used in [7], there are 73472 pieces of sentences after processing, and the vocabulary contains 22234 words. While our adopted ReVerb45K dataset has at least 89055 pieces of sentences and 39834 words of vocabulary.

Figure 7 shows the relationship between the sentence similarity score and the SNR over AWGN, Rayleigh, and Rician channels. While Figure 8 shows the relationship between the WER score and the SNR over AWGN, Rayleigh, and Rician channels. It can be seen that when using sentence similarity and WER as the evaluation metric, our proposed KESC-T model outperforms DeepSC in all channel conditions, and KESC-T also outperforms the traditional method in low-SNR scenarios. It further confirms that the performance of semantic communication can be significantly improved by adding a

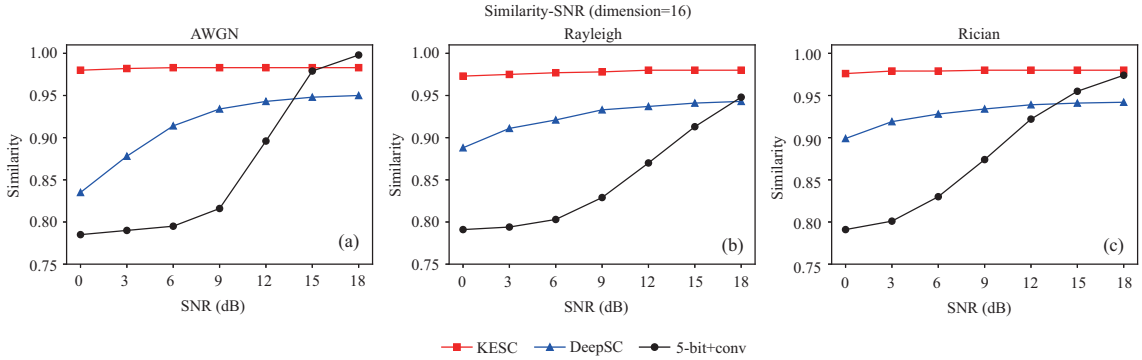

Figure 7 (Color online) The sentence similarity score versus SNR in (a) AWGN, (b) Rayleigh, and (c) Rician channels.

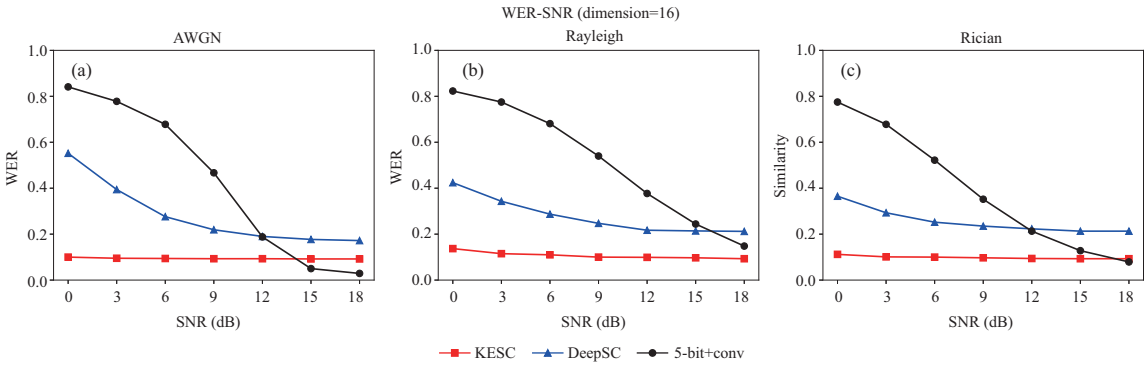

Figure 8 (Color online) The WER score versus SNR in (a) AWGN, (b) Rayleigh, and (c) Rician channels.

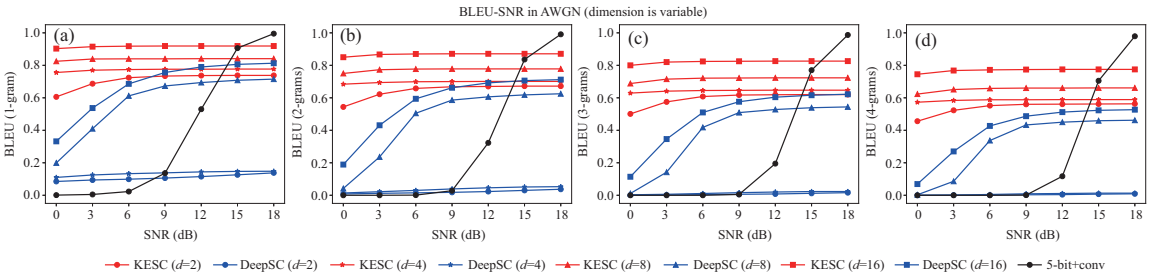

Figure 9 (Color online) The BLEU score versus SNR with different vector dimensions. (a) 1-gram BLEU, (b) 2-grams BLEU, (c) 3-grams BLEU, and (d) 4-grams BLEU.

knowledge base.

To reduce the number of semantic symbols transmitted in the communication process, we turn to reduce the number of neurons in the hidden layer of the autoencoder. By compressing the semantic vector into the lower dimension, we obtain the performance of our proposed KESC-T model under different dimensions of the hidden layer. Figure 9 shows the relationship between the BLEU score and the SNR over the AWGN channel with different dimensions of the hidden layer. It can be seen that the performance of our proposed KESC-T model is better than the DeepSC model. As the dimension of the output semantic vector decreases, the performance of the KESC-T model degrades, but the degradation is much slower than that of the DeepSC model. The receiver in our proposed KESC system can still meet the requirements of semantic communication in resource-constrained scenarios. However, the DeepSC model can recover semantic information only when the semantic vectors achieve 8 or 16 dimensions, and cannot work at the state with the semantic vectors compressed to 2 or 4 dimensions.

Figure 10 shows the relationship between the BLEU score and the SNR over AWGN channels with different batch sizes. It can be seen that the performance of our proposed KESC-T gradually improves as the training batch size increases. This implies that a larger batch size is more beneficial for the model to obtain overall information, as well as reducing the probability of achieving the local optimum point during the training process. However, better performance is at the cost of GPU memory. It is necessary

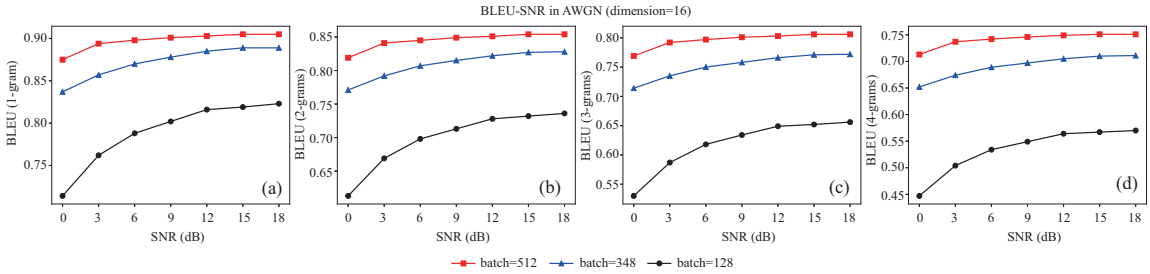

Figure 10 (Color online) The BLEU score versus SNR with different batch sizes. (a) 1-gram BLEU, (b) 2-grams BLEU, (c) 3-grams BLEU, and (d) 4-grams BLEU.

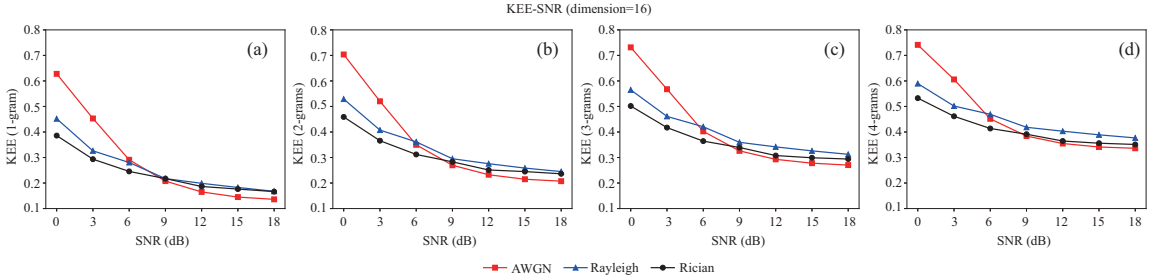

Figure 11 (Color online) The KEE score versus SNR in different channels. (a) 1-gram BLEU, (b) 2-grams BLEU, (c) 3-grams BLEU, and (d) 4-grams BLEU.

to trade off the semantic performance and GPU consumption.

Figure 11 shows the relationship between the KEE score and the SNR over AWGN, Rayleigh, and Rician channels. With the increase of SNR, the knowledge utilization efficiency of the model decreases, which confirms the applicability of our proposed model in low-SNR scenarios. In addition, we can see that as the value of n in the n-gram increases, the KEE score of the model also increases. It indicates that the input of external knowledge has a great contribution to the recovery of consecutive words.

7 Conclusion

In this paper, we propose a KESC system named KESC which uses OFDM as the transmission method. A cloud-edge-device collaboration framework is proposed to support the storage of the knowledge base, which consists of a knowledge graph to improve the performance of semantic communication. Furthermore, we formulate the semantic pilot to transmit external knowledge and design the corresponding transmission mechanism. To realize KESC, we propose a multi-encoder transformer-based neural network model, which uses the semantic vector sent by the transmitter and the knowledge embedding in the knowledge base simultaneously for auxiliary semantic restoration. In addition, we define a performance metric named KEE to evaluate the efficiency of the knowledge graph. The proposed KESC system is evaluated by four different performance metrics, including BLEU, sentence similarity, WER, and KEE. The results validate that the performance of our proposed KESC-T model outperforms the compared DeepSC model. Moreover, our proposed KESC-T model obtains better performance under low-SNR and resource-constrained scenarios.

Acknowledgements This work was supported in part by Key R&D Program of Shandong Province (Grant No. 2020CXGC010109), National Natural Science Foundation of China (Grant No. 62201079), Fundamental Research Funds for the Central Universities (Grant No. 2022RC15), and Major Key Project of PCL.

References

1 You X, Wang C X, Huang J, et al. Towards 6G wireless communication networks: vision, enabling technologies, and new paradigm shifts. Sci China Inf Sci, 2021, 64: 110301
2 Calvanese Strinati E, Barbarossa S. 6G networks: beyond Shannon towards semantic and goal-oriented communications. Comput Netw, 2021, 190: 107930
3 Popovski P, Simeone O, Boccardi F, et al. Semantic-effectiveness filtering and control for post-5G wireless connectivity. J Ind Inst Sci, 2020, 100: 435–443
4 Juba B, Sudan M. Universal semantic communication II: a theory of goal-oriented communication. In: Proceedings of Electronic Colloquium on Computational Complexity (ECCC), 2008

5 Goldreich O, Juba B, Sudan M. A theory of goal-oriented communication. J ACM, 2012, 59: 1–65
6 Zhang Y C, Zhang P, Wei J B, et al. Semantic communication for intelligent devices: architectures and a paradigm (in Chinese). Sci Sin Inform, 2022, 52: 907–921
7 Xie H, Qin Z, Li G Y, et al. Deep learning enabled semantic communication systems. IEEE Trans Signal Process, 2021, 69: 2663–2675
8 Papineni K, Roukos S, Ward T, et al. BLEU: a method for automatic evaluation of machine translation. In: Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics, 2002. 311–318
9 Klakow D, Peters J. Testing the correlation of word error rate and perplexity. Speech Commun, 2002, 38: 19–28
10 Xie H, Qin Z. A lite distributed semantic communication system for Internet of Things. IEEE J Sel Areas Commun, 2021, 39: 142–153
11 Weng Z, Qin Z, and Li G Y. Semantic communications for speech signals. In: Proceedings of ICC 2021-IEEE International Conference on Communications, 2021. 1–6
12 Xie H, Qin Z, Li G Y. Task-oriented multi-user semantic communications for VQA. IEEE Wirel Commun Lett, 2022, 11: 553–557
13 Shi G M, Gao D H, Song X D, et al. A new communication paradigm: from bit accuracy to semantic fidelity. 2021. ArXiv:2101.12649
14 Shi G, Xiao Y, Li Y, et al. From semantic communication to semantic-aware networking: model, architecture, and open problems. IEEE Commun Mag, 2021, 59: 44–50
15 Zhang P, Xu W, Gao H, et al. Toward wisdom-evolutionary and primitive-concise 6G: a new paradigm of semantic communication networks. Engineering, 2022, 8: 60–73
16 Ji S, Pan S, Cambria E, et al. A survey on knowledge graphs: representation, acquisition, and applications. IEEE Trans Neural Netw Learn Syst, 2022, 33: 494–514
17 Zhang Z, Han X, Liu Z, et al. ERNIE: enhanced language representation with informative entities. In: Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, 2019. 1441–1451
18 Yang M, Bian C, Kim H S. Deep joint source channel coding for wireless image transmission with OFDM. In: Proceedings of ICC 2021-IEEE International Conference on Communications, 2021. 1–6
19 Weaver W. Recent contributions to the mathematical theory of communication. ETC Rev Gen Semant, 1953, 10: 261–281
20 Bao J, Basu P, Dean M, et al. Towards a theory of semantic communication. In: Proceedings of IEEE Network Science Workshop, 2011. 110–117
21 Zhang P, Xu X, Dong C, et al. Intellicise communication system: model-driven semantic communications. J China Univ Post Telecommun, 2022, 29: 2
22 Zhou F, Li Y, Zhang X, et al. Cognitive semantic communication systems driven by knowledge graph. In: Proceedings of IEEE International Conference on Communications, 2022
23 Yang W, Liew Z Q, Lim W Y B, et al. Semantic communication meets edge intelligence. IEEE Wirel Commun, 2022, 29: 28–35
24 Devlin J, Chang M, Lee K, et al. BERT: pre-training of deep bidirectional transformers for language understanding. In: Proceedings of NAACL-HLT, 2019. 4171–4186
25 Liu W, Zhou P, Zhao Z, et al. K-bert: enabling language representation with knowledge graph. In: Proceedings of the AAAI Conference on Artificial Intelligence, 2020. 2901–2908
26 Liu Y, Wan Y, He L, et al. KG-BART: knowledge graph-augmented bart for generative commonsense reasoning. In: Proceedings of the AAAI Conference on Artificial Intelligence, 2021. 6418–6425
27 Lu Y, Zhang J, Zong C. Exploiting knowledge graph in neural machine translation. In: Proceedings of China Workshop on Machine Translation, 2018. 27–38
28 Moussallem D, Arčan M, Ngomo A C N, et al. Augmenting neural machine translation with knowledge graphs. 2019. ArXiv:1902.08816
29 Zhao Y, Xiang L, Zhu J, et al. Knowledge graph enhanced neural machine translation via multi-task learning on sub-entity granularity. In: Proceedings of the 28th International Conference on Computational Linguistics, 2020. 4495–4505
30 Zhao Y, Zhang J, Zhou Y, et al. Knowledge graphs enhanced neural machine translation. In: Proceedings of the 29th International Conference on International Joint Conferences on Artificial Intelligence, 2021. 4039–4045
31 Xie S, Xia Y, Wu L, et al. End-to-end entity-aware neural machine translation. Mach Learn, 2022, 111: 1181–1203
32 Li B, Liu H, Wang Z, et al. Does multi-encoder help? A case study on context-aware neural machine translation. In: Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, 2020. 3512–3518
33 Libovickỳ J, Helcl J, Mareček D. Input combination strategies for multi-source transformer decoder. In: Proceedings of the 3rd Conference on Machine Translation: Research Papers, 2018
34 Shin J, Lee J H. Multi-encoder transformer network for automatic post-editing. In: Proceedings of the 3rd Conference on Machine Translation: Shared Task Papers, 2018. 840–845
35 Li Y, Feng R, Rehg I, et al. Transformer-based neural text generation with syntactic guidance. 2020. ArXiv:2010.01737
36 Lohrenz T, Li Z, Fingscheidt T. Multi-encoder learning and stream fusion for transformer-based end-to-end automatic speech recognition. 2021. ArXiv:2104.00120
37 Lin T, Wang Y, Liu X, et al. A survey of transformers. AI Open, 2022, 3: 111–132
38 Vaswani A, Shazeer N, Parmar N, et al. Attention is all you need. In: Proceedings of the 31st International Conference on Neural Information Processing Systems, 2017
39 Vashishth S, Jain P, Talukdar P. CESI: canonicalizing open knowledge bases using embeddings and side information. In: Proceedings of World Wide Web Conference, 2018. 1317–1327
40 Fader A, Soderland S, Etzioni O. Identifying relations for open information extraction. In: Proceedings of Conference on Empirical Methods in Natural Language Processing, 2011. 1535–1545
41 Callan J, Hoy M, Yoo C, et al. Clueweb09 data set. 2009. https://lemurproject.org/clueweb09/index.php
42 Pennington J, Socher R, Manning C D. GloVe: global vectors for word representation. In: Proceedings of Conference on Empirical Methods in Natural Language Processing (EMNLP), 2014. 1532–1543