3.2 Commonsense Knowledge Injection

As discussed above, understanding complex clauses requires ECPE models to be equipped with external commonsense knowledge. To this end, we adopt the well-known sememe knowledge graph HowNet [5] to enrich the semantic information of the input documents. Due to its significance in understanding the nature of semantics in human languages, HowNet has been widely used in various natural language processing tasks such as sentiment analysis and word sense disambiguation. In HowNet, each Chinese word is annotated with its minimum semantic units called sememes, and each word corresponds to a sememe set. For example, `apple` is associated with two meanings (i.e., brand and fruit), and it has a list of sememes including {`computer`, `able`, `bring`, $\cdots$}.

Formally, given an input clause $c_i = \{w_i^1, w_i^2, \cdots, w_i^{|c_i|}\}$, we extract the sememes of each word (if exist) in HowNet. These extracted sememes are considered as external knowledge that helps to understand the semantics of input documents. We then inject these knowledge into the original clause c_i and obtain the expanded clause $c_i' = [w_i^1, w_i^2, \cdots, w_i^j \cup \{e_i^{j,k}\}_{k=1}^K, \cdots, w_i^{|c_i|}]$, where $\{e_i^{j,k}\}_{k=1}^K$ is the sememe set of word w_i^j in the KG. Since too many additional entities may cause noisy information, we restrict the maximum number of added entities to K, and $K = 2$ works well. Figure 3 shows an example of the expanded clause.

Fig. 3. The process of commonsense knowledge injection.

3.3 Knowledge-Enhanced Clause Encoding

To learn knowledge-enriched clause representations, we propose a hierarchical Transformers, which consists of two components: a word-level Transformer and a clause-level Transformer to integrate the injected knowledge and document information at both local and global stages.

Word-Level Transformer. Word-level transformer aims to learn a knowledge-enrich representation for each clause by encoding commonsense knowledge and semantic information of clauses. Different from the vanilla Transformer [13], our word-level Transformer introduces two encoding schemes: a position encoding to model the position information of the original clauses and the added knowledge, and a flag encoding to enable the model to pay more attention to emotion words.

Specifically, given an expanded clause c_i' consisting of words and relevant entities, our positional encoding is to assign each token a position index to characterize the relative position of the injected knowledge entities in the clause. If the token corresponds to a word $w_i^j \in c_i$, its position index is the superscript j of the word. If the word w_i^j can find corresponding knowledge entities $\{e_i^{j,k}\}_{k=1}^K$, their position indexes are the superscript of the start character in its associated word w_i^j. Formally, the position indices p_i of clause c_i'

are $p_i = [1, 2, \cdots, j \cup [j]_{\times K}, \cdots, |c_i|]$, where $[j]_{\times K}$ denotes a sequence containing K identical indices. Figure 3 illustrates an example of our proposed positional encoding. The clause is expanded by two knowledge entities: `fall` and `dangerous`, which are extracted from commonsense KG based on the word `teetering`. The entities are assigned the same index as the word `teetering`, and therefore its semantic information can be aggregated equally by the word-level Transformer.

Emotion words in clauses play an important role in modeling the semantic information of clauses. Based on this idea, we introduce a flag encoding to highlight the importance of emotion words (all emotion words have already been annotated in the dataset). The flag encoding is to assign 0 or 1 for each word to indicate whether the token is an emotion word. This is formalized as:

$$f_i^j = \begin{cases} 1, & x_i^j \in E \\ 0, & x_i^j \notin E \end{cases}, \quad x_i^j \in c_i', \tag{1}$$

where E is the set of emotion words.

For each token in c_i', we construct its input embedding z_j by summing its element embedding, position embedding, and flag embedding as $z_j = x_j^{ele} + x_j^{pos} + x_j^{flg}$, where x_j^{ele} is the element embedding of the j-th token in the clause c_i', x_j^{pos} is the position embedding corresponding to its position index and x_j^{flg} is the flag embedding corresponding to its flag index. So the clause c_i' can be represented as $Z_{c_i'}^0 = [z_1, \cdots, z_i, \cdots]$, which will be fed into the word-level Transformer.

Similar to the original Transformer, the word-level Transformer consists of L stacked transformer blocks. The l-th transformer block takes z_i^l as the input and outputs the hidden representation z_i^{l+1} of the l-th layer about the word w_j^i, which is the input sequence of the $l+1$ transformer block. The calculation process is as follows:

$$\begin{aligned} \hat{Z}_{c_i'}^l &= \mathrm{LN}(Z_{c_i'}^l + \mathrm{MHA}(Z_{c_i'}^l)), \\ Z_{c_i'}^{l+1} &= \mathrm{LN}(\hat{Z}_{c_i'}^l + \mathrm{MLP}(\hat{Z}_{c_i'}^l)), \end{aligned} \tag{2}$$

where LN is the Layer-Norm unit, and MLP is the Multi-layer Perceptron. We leverage Multi-Head Attention (MHA) to encode the input representations of words, which can be formalized as:

$$\mathrm{MHA}(Z_{c_i'}^l) = W^O \Big\|_{k=1}^{H} \mathrm{ATT}_i(Z_{c_i'}^l W_k^Q, Z_{c_i'}^l W_k^K, Z_{c_i'}^l W_k^V), \tag{3}$$

where $W_k^Q, W_k^K, W_k^V, W^O \in \mathbb{R}^{d \times d}$ are learnable weight matrices; d is the embedding dimension; k denotes the number of attention heads; $\|$ is the concatenation operation. For i-th attention head, the scale dot-product is as follows:

$$\begin{aligned} &\mathrm{ATT}_i(Z_{c_i'}^l W_k^Q, Z_{c_i'}^l W_k^K, Z_{c_i'}^l W_k^V) \\ &= \mathrm{softmax}\left(\frac{Z_{c_i'}^l W_k^Q (Z_{c_i'}^l W_k^K)^T}{\sqrt{d}}\right) Z_{c_i'}^l W_k^V. \end{aligned} \tag{4}$$

Finally, we can obtain the clause representation h_i of the clause c'_i by averaging the hidden state of the final layer word representations:

$$h_i = \frac{1}{|c'_i|} \sum_{j=1}^{|c'_i|} z_j^L, \tag{5}$$

where $|c'_i|$ is the number of tokens in the expanded clause c'_i.

Clause-level Transformer. In this part, we aim to model inter-clause relationships and learn interactive representations of clauses. Specifically, we formalize the clause-level representation of the document D as $H_D = \{h_1, \cdots, h_i, \cdots\}$, where h_i is the knowledge-enriched representation of the i-th clause in D. We use the absolute position to formalize its position indices as: $[1, 2, \cdots, |D|]$. For each clause in D, we construct its input embedding s_i by summing its element embedding and position embedding as $s_i = h_i^{ele} + h_i^{pos}$, where h_i^{ele} is h_i and h_i^{pos} is the position embedding corresponding to its position index. The document D can be represented as $S_D = \{s_1, \cdots, s_i, \cdots, s_{|D|}\}$, which will be fed into the clause-level Transformer. The clause-level Transformer consists of F stacked transformer blocks, which can be formalized as:

$$\begin{aligned} \mathrm{MHA}(S_D^f) &= W^P \Big\|_{m=1}^{H} \mathrm{ATT}_i(S_D^f W_m^Q, S_D^f W_m^K, S_D^f W_m^V), \\ \hat{S}_D^f &= \mathrm{LN}(S_D^f + \mathrm{MHA}(S_D^f)), \\ S_D^{f+1} &= \mathrm{LN}(\hat{S}_D^f + \mathrm{MLP}(\hat{S}_D^f)), \end{aligned} \tag{6}$$

where $W_m^Q, W_m^K, W_m^V, W^P \in \mathbb{R}^{d \times d}$ are trainable weight matrices, and $\|$ denotes the concatenation operation. For i-th attention head, the scale dot-product is performed as:

$$\begin{aligned} &\mathrm{ATT}_i(S_D^f W_m^Q, S_D^f W_m^K, S_D^f W_m^V) \\ &= \mathrm{softmax}\left(\frac{S_D^f W_m^Q (S_D^f W_m^K)^T}{\sqrt{d}}\right) S_D^f W_m^V. \end{aligned} \tag{7}$$

Finally, we couple the original clause representations and the weighted clause representations. The final clause representations can be formulated as:

$$\boldsymbol{O}_D = S_D + S_D^F W_D, \tag{8}$$

where $W_D \in \mathbb{R}^{d \times d}$ is a learnable weight matrix; $\boldsymbol{O}_D = \{\boldsymbol{o}_1, \cdots, \boldsymbol{o}_i, \cdots\}$, and $\boldsymbol{o}_i$ is the final clause representation of the clause c_i, which integrates both the semantic information and the inter-clause relationships in the document.

After obtaining the final clause representations, we use two MLP layers to predict whether each clause is an emotion/cause clause or none of both:

$$\hat{y}_i^{emo} = \sigma(\boldsymbol{o}_i W_{emo} + b_{emo}), \tag{9}$$

$$\hat{y}_i^{cau} = \sigma(\boldsymbol{o}_i W_{cau} + b_{cau}), \tag{10}$$

where W_{emo} , $W_{cau} \in \mathbb{R}^{d \times 1}$ and b_{emo}, $b_{cau} \in \mathbb{R}$ are weight matrixes and bias vectors for the emotion prediction MLP layer and cause prediction MLP layer respectively, and $\sigma(\cdot)$ is the logistic function.

3.4 Emotion-Cause Pair Extraction

For all clauses from document $D = \{c_1, c_2, \cdots, c_{|D|}\}$, we consider the combination of all clauses. As a result, there are $|D| * |D|$ possible cases of clause pairs, which can be formalized as $\mathcal{P} = \{(c_1, c_1), \cdots, (c_i, c_j), \cdots, (c_{|D|}, c_{|D|})\}$, where $\mathcal{P}$ is the set of all candidate pairs, and (c_i, c_j) means that cause clause c_j and emotion clause c_i can formulate a candidate pair. For each candidate clause pair (c_i, c_j), we can get their clause pair representation $\boldsymbol{p}_{ij}$, as follows:

$$\boldsymbol{p}_{ij} = ReLU([\boldsymbol{o}_i; \boldsymbol{o}_j; r_{j-i}]W_p + b_p), \tag{11}$$

where W_p and b_p are learnable weight matrix and bias vector, $\boldsymbol{o}_i$ and $\boldsymbol{o}_j$ represents the clause c_i, c_j embedding respectively, r_{j-i} represents the relative position embedding of clause pair (c_i, c_j). For each relative position $\beta \in \{-k, \cdots, +k\}$, its embeddings r_m can be obtained by:

$$r_\beta = \sum_{j=-k}^{+k} exp(-(j-\beta)^2) \cdot r'_j, \tag{12}$$

where $j \in \{-k, \cdots, +k\}$ denotes one of all possible relative position values, and r'_j is randomly intialized via sampling from a uniform distribution.

Then we use a MLP layer to predict the score of each candidate pair $\boldsymbol{p}_{ij}$:

$$\hat{y}_{ij} = tanh\left(\boldsymbol{p}_{ij}W_s + b_s\right), \tag{13}$$

where $W_s \in \mathbb{R}^{d\times 1}$ and $b_s \in \mathbb{R}$ are weight matrix and bias vector respectively.

Our method is trained in an end-to-end manner. We use a cross-entropy loss function $\mathcal{L}_{pair}$ for the emotion-cause pair extraction task, and two cross-entropy loss functions $\mathcal{L}_{emo}$ and $\mathcal{L}_{cau}$ for two sub-tasks: emotion clause extraction task and cause clause extraction task, respectively. Finally, the objective function can be defined as the sum of the above loss functions, as follows:

$$\mathcal{L} = \mathcal{L}_{pair} + \lambda(\mathcal{L}_{emo} + \mathcal{L}_{cau}). \tag{14}$$

where hyper-parameter λ controls the tradeoff between emotion-cause pair extraction and sub-tasks: emotion clause extraction and cause clause extraction.

4 Experiments

4.1 Datasets and Metrics

We conduct our experiments on the benchmark dataset which is released by Xia and Ding [17]. The dataset is constructed based on an emotion cause extraction corpus [8] which consists of 1,945 Chinese documents. We summarize the detailed statistics in Table 1. We repeat the experiments 15 times, and report the average results of precision (P), recall (R), and F1-score (F1) on the main task: emotion-cause pair extraction (ECPE) and two sub-tasks: emotion clause extraction (EE) and cause clause extraction (CE).

Table 1. Statistics of the dataset. "Doc." is the abbreviation for "Document".

# Doc. with one emotion-cause pair	1746
# Doc. with two emotion-cause pairs	177
# Doc. with three or more emotion-cause pairs	22
Avg. of clauses per document	14.77
Max. of clauses per document	77

4.2 Baselines

To confirm the effectiveness of our method, we compare our method with following baseline methods. **Indep** [17] is a two-step method that extracts emotion clauses and cause clauses respectively, then pair them and select the final emotion-cause pairs. **Inter-CE** [17] is an variant of Indep which uses emotion extraction to improve cause extraction. **Inter-EC** [17] is an variant of Indep which uses cause extraction to improve emotion extraction. **RankCP** [15] is a unified framework to tackle ECPE task from a ranking perspective. **ECPE-2D** [3] represents the emotion-cause pairs by a 2D representation scheme, and integrates the emotion-cause pair representation, interaction and prediction into a joint framework. **PTN** [16] is a neural network which tackles the complete emotion-cause pair extraction in one unified tagging task. **MGSAG** [1] aims to alleviate the position bias problem by incorporating fine-grained and coarse-grained semantic features jointly. **MaCa** [19] extracts emotion-cause pairs as a procedure of sequence modeling.

4.3 Implementation Details

In this paper, we adopt $\text{BERT}_{Chinese}$ as the basis in this work. For both word-level and clause-level Transformer, we set the layer number of each transformer block to 1, attention heads to 1, the embedding size to 768, the hidden size to 3072, and we add dropout with the rate of 0.1 to reduce overfitting.. The dimension of relative position embedding is set to 50 and the maximum relative position value k to 12. Moreover, we use the Adam optimizer [10] as the optimizer with an initial learning rate of $3e-5$. We implement our method and further conduct it on a server with 2 NVIDIA GeForce RTX 3090 GPUs.

4.4 Comparison with ECPE Methods

Table 2 reports the comparative results on the ECPE task and two sub-tasks, i.e., EE and CE. To be fair, here we adopt the experimental results of all methods using the BERT model (if the model uses the BERT), and all experimental results are obtained from the references. From the table, we can observe that:

– Compared to other ECPE baselines, KEHT achieves the best performance. In particular, for the ECPE task, KEHT improves over the best baseline by 2.58% for F_1 score, and 4.32% for R score. For the EE task, KEHT improves

Table 2. Comparison of our method with other baseline methods.

	ECPE			EE			CE		
	F_1	P	R	F_1	P	R	F_1	P	R
Indep	0.5818	0.6832	0.5082	0.8210	0.8375	0.8071	0.6205	0.6902	0.5673
Inter-CE	0.5901	0.6902	0.5135	0.8300	0.8494	0.8122	0.6151	0.6809	0.5634
Inter-EC	0.6128	0.6721	0.5705	0.8230	0.8364	0.8107	0.6507	0.7041	0.6083
RankCP	0.7360	0.7119	0.7630	0.9057	0.9123	0.8999	0.7615	0.7461	0.7788
ECPE-2D	0.6889	0.7292	0.6544	0.8910	0.8627	0.9221	0.7123	0.7336	0.6934
PTN	0.6650	0.7600	0.5918	0.8360	0.8447	0.8278	0.6799	0.7175	0.6470
MGSAG	0.7521	0.7743	0.7321	0.8717	0.9208	0.9211	0.7712	**0.7979**	0.7568
MaCa	0.7387	**0.8047**	0.7215	0.8704	0.8819	0.8955	0.7435	0.7841	0.7260
Ours	**0.7779**	0.7526	**0.8062**	**0.9455**	**0.9481**	**0.9431**	**0.7858**	0.7672	**0.8068**

Table 3. Comparison of different supervised signals for KEHT.

Loss Function	F_1	P	R
$\mathcal{L}_{pair}$	0.7545	0.7291	0.7831
$\mathcal{L}_{pair} + \lambda(\mathcal{L}_{emo} + \mathcal{L}_{cau})$	**0.7779**	**0.7526**	**0.8062**

over the best baseline by 3.98% for F_1 score, 2.73% for P score, and 2.1% for R score. For the CE task, KEHT still outperforms other baselines by 1.46% for F_1 score, and 2.8% for the R score. The experimental results demonstrate the effectiveness of utilizing external commonsense knowledge to build knowledge-enhanced texts for extracting emotion-cause pair.

– KEHT outperforms other ECPE methods on the ECPE task and two sub-tasks. The main reason would be that other methods only extract emotion-cause pairs by directly analyzing the documents on the basis of a large training set and this is insufficient to model the complex relations between clauses. On the contrary, our KEHT utilizes external commonsense knowledge to build knowledge-enhanced texts and encode texts at both word level and clause level. We can also observe that MGSAG outperforms us on the P score of the CE task and MaCa outperforms us on the P score of the ECPE task. The main reason would be that MGSAG and MaCa may abandon some true emotion-cause pairs.

Effect of Two-Level Supervision. We use two-level supervised signals to train our method: a low-level signal $\mathcal{L}_{emo} + \mathcal{L}_{cau}$ at the output of the clause-level Transformer (see Eq. 9), and a high-level signal $\mathcal{L}_{pair}$ at the classification stage (see Eq. 13). Table 3 shows the effect of two-level supervision. The results show that training with two-level supervision improves extraction performance. This indicates that two-level supervision is helpful for learning clause representations and facilitates the emotion-cause pair extraction process.

Effect of Commonsense Knowledge Injection. We remove the knowledge graph which is used for injecting external commonsense knowledge to construct the knowledge-enhanced texts. Figure 4 reports that method without external knowledge results in a drop. The F_1 decreases 1.49%, and P, R drop 1.36% and

1.22%, respectively. It shows that utilizing external commonsense knowledge can enrich semantic information and significantly improve extraction performance.

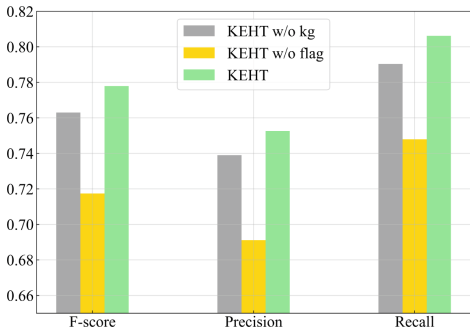

Fig. 4. Comparative results of KEHT which removes different components

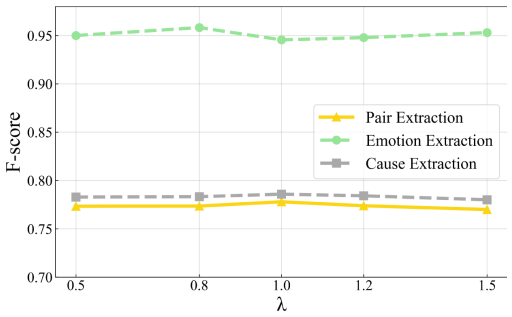

Fig. 5. Results with various values of λ.

Effect of Flag Embedding. We further investigate the importance of flag embedding. Figure 4 shows that by removing the flag embedding part in KEHT, the performance results in a significant drop. The F_1 decreases 6.05%, and P, R drop 6.15% and 5.83%, respectively. We can conclude that flag embedding is useful for our method to focus on more important information.

Effect of Hyperparameter for Loss Function. To explore the influence of two sub-tasks, we conduct a deeper analysis of the hyperparameter λ which is used in Eq. 14. We vary the values of λ from 0.5 to 1.5, and the results are shown in Fig. 5. We can observe that with the increase of λ, the performance of our method increases and then decreases slightly, and reaches a peak value at $\lambda = 1$ on the emotion-cause pair extraction and the cause clause extraction and at $\lambda = 0.8$ on the emotion clause extraction. This suggests that the emotion-cause pair extraction task and two sub-tasks play an equally important role.

5 Conclusion and Future Work

In this paper, we propose a novel KEHT method to tackle the emotion-cause pair extraction task. Our method effectively incorporates external commonsense knowledge into the clause representations via a hierarchical Transformers module, which leverages two different types of transformer blocks to encode knowledge-enriched clause representations at both global and local stages. In future work, we would like to tackle the problem of the documents with more than one emotion-cause pair by exploring the more fine-grained roles of the clauses.

Acknowledgements. This work is supported by the National Key Research and Development Program of China (under grant 2020AAA0106100).

References

1. Bao, Y., Ma, Q., Wei, L., Zhou, W., Hu, S.: Multi-granularity semantic aware graph model for reducing position bias in emotion cause pair extraction. In: Findings of ACL, pp. 1203–1213 (2022)
2. Cheng, Z., Jiang, Z., Yin, Y., Li, N., Gu, Q.: A unified target-oriented sequence-to-sequence model for emotion-cause pair extraction. IEEE/ACM Trans. Audio Speech Lang. Process. **29**, 2779–2791 (2021)
3. Ding, Z., Xia, R., Yu, J.: ECPE-2D: emotion-cause pair extraction based on joint two-dimensional representation, interaction and prediction. In: ACL, pp. 3161–3170 (2020)
4. Ding, Z., Xia, R., Yu, J.: End-to-end emotion-cause pair extraction based on sliding window multi-label learning. In: EMNLP, pp. 3574–3583 (2020)
5. DongZ, D., HAO, C.: Hownet and the computation of meaning (2006)
6. Fan, C., et al.: A knowledge regularized hierarchical approach for emotion cause analysis. In: EMNLP-IJCNLP, pp. 5614–5624 (2019)
7. Fan, C., Yuan, C., Du, J., Gui, L., Yang, M., Xu, R.: Transition-based directed graph construction for emotion-cause pair extraction. In: ACL, pp. 3707–3717 (2020)
8. Gui, L., Xu, R., Wu, D., Lu, Q., Zhou, Y.: Event-driven emotion cause extraction with corpus construction. In: EMNLP, pp. 145–160. World Scientific (2018)
9. Huang, W., Yang, Y., Peng, Z., Xiong, L., Huang, X.: Deep neural networks based on span association prediction for emotion-cause pair extraction. Sensors **22**(10), 3637 (2022)
10. Kingma, D.P., Ba, J.: Adam: a method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
11. Mittal, A., Vaishnav, J.T., Kaliki, A., Johns, N., Pease, W.: Emotion-cause pair extraction in customer reviews. arXiv preprint arXiv:2112.03984 (2021)
12. Turcan, E., Wang, S., Anubhai, R., Bhattacharjee, K., Al-Onaizan, Y., Muresan, S.: Multi-task learning and adapted knowledge models for emotion-cause extraction. arXiv preprint arXiv:2106.09790 (2021)
13. Vaswani, A., et al.: Attention is all you need. In: Advances in Neural Information Processing Systems, vol. 30 (2017)
14. Wang, F., Ding, Z., Xia, R., Li, Z., Yu, J.: Multimodal emotion-cause pair extraction in conversations. arXiv preprint arXiv:2110.08020 (2021)
15. Wei, P., Zhao, J., Mao, W.: Effective inter-clause modeling for end-to-end emotion-cause pair extraction. In: ACL, pp. 3171–3181 (2020)
16. Wu, Z., Dai, X., Xia, R.: Pairwise tagging framework for end-to-end emotion-cause pair extraction. Front. Comp. Sci. **17**(2), 1–10 (2023)
17. Xia, R., Ding, Z.: Emotion-cause pair extraction: a new task to emotion analysis in texts. In: ACL, pp. 1003–1012 (2019)
18. Yan, H., Gui, L., Pergola, G., He, Y.: Position bias mitigation: a knowledge-aware graph model for emotion cause extraction. arXiv preprint arXiv:2106.03518 (2021)
19. Yang, C., Zhang, Z., Ding, J., Zheng, W., Jing, Z., Li, Y.: A multi-granularity network for emotion-cause pair extraction via matrix capsule. In: CIKM, pp. 4625–4629 (2022)
20. Yuan, C., Fan, C., Bao, J., Xu, R.: Emotion-cause pair extraction as sequence labeling based on a novel tagging scheme. In: EMNLP, pp. 3568–3573 (2020)

PICKD: In-Situ Prompt Tuning for Knowledge-Grounded Dialogue Generation

Rajdeep Sarkar[1(✉)], Koustava Goswami[2], Mihael Arcan[1], and John McCrae[1]

[1] University of Galway, Galway, Ireland
{r.sarkar1,mihael.arcan,john.mccrae}@universityofgalway.ie
[2] Adobe Research Bangalore, Bangalore, India
koustavag@adobe.com

Abstract. Generating informative, coherent and fluent responses to user queries is challenging yet critical for a rich user experience and the eventual success of dialogue systems. Knowledge-grounded dialogue systems leverage external knowledge to induce relevant facts in a dialogue. These systems need to understand the semantic relatedness between the dialogue context and the available knowledge, thereby utilising this information for response generation. Although various innovative models have been proposed, they neither utilise the semantic entailment between the dialogue history and the knowledge nor effectively process knowledge from both structured and unstructured sources. In this work, we propose PICKD, a two-stage framework for knowledgeable dialogue. The first stage involves the *Knowledge Selector* choosing knowledge pertinent to the dialogue context from both structured and unstructured knowledge sources. PICKD leverages novel *In-Situ* prompt tuning for knowledge selection, wherein prompt tokens are injected into the dialogue-knowledge text tokens during knowledge retrieval. The second stage employs the *Response Generator* for generating fluent and factual responses by utilising the retrieved knowledge and the dialogue context. Extensive experiments on three domain-specific datasets exhibit the effectiveness of PICKD over other baseline methodologies for knowledge-grounded dialogue. The source is available at https://github.com/rajbsk/pickd.

Keywords: Language Model Prompting · Knowledge grounded Dialogue Systems · Knowledge Graphs

1 Introduction

With the proliferation of personal assistants (Siri, Alexa, etc.), research on dialogue systems has gained a lot of traction. Inducing relevant information in responses leads to a fluent, engaging and coherent conversation with a dialogue system. While language models help develop fluent dialogue systems [16], such systems lack the necessary tools for generating accurate responses. Researchers have utilised external knowledge from either unstructured knowledge sources such as Wikipedia articles [1,13], domain-grounded documents [14] or structured sources like Knowledge Graphs (KGs) [19,22] for generating informative responses.

H. Kashima et al. (Eds.): PAKDD 2023, LNAI 13938, pp. 124–136, 2023.
https://doi.org/10.1007/978-3-031-33383-5_10

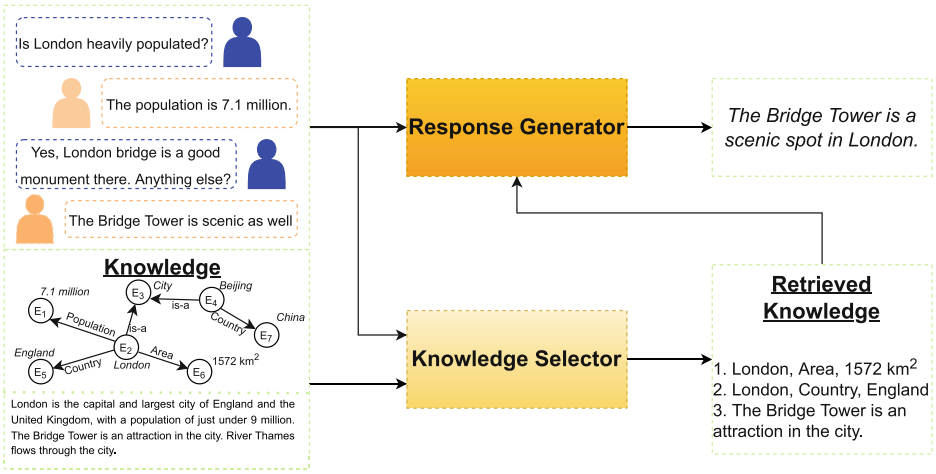

Fig. 1. Modular overview of the PICKD framework. The *Knowledge Selector* retrieves dialogue context-relevant knowledge from the structured and unstructured knowledge available. The retrieved knowledge and along with the dialogue context is then sent to the *Response Generator* for producing informative and fluent responses.

Regardless of the recent improvements in generic dialogue systems, the systems must interpret the semantic relatedness between the dialogue context and the external knowledge. For example, in Fig. 1, while responding to utterance 1, the model needs to understand the context encompasses "London" and then utilise the knowledge triple "London, has population, 7.1 million" when generating the response text. Similarly, when responding to utterance 3, the agent needs to register the contextual overlap between the dialogue history and the knowledge sentence "The London Bridge and the Bridge Tower are major attractions in London.". It should then leverage this information while generating utterance 4.

While prior works have posited novel frameworks to address this task, they tend to cater exclusively to unstructured knowledge sources like paragraphs/documents or solely to structured sources like KGs. Zhou et al. [21] proposed the KdConv dataset, which spans three domains and includes both structured and unstructured knowledge. They also proposed adapting Seq2Seq and the Hierarchical Recurrent Encoder-Decoder Framework (HRED) with knowledge memory for response generation. Wang et al. [11] built upon this work with their RT-KGD framework, which uses a Heterogeneous Graph Transformer Network to capture information flow and BART [4] for response generation. However, these models do not represent dialogue context and knowledge elements in the same semantic space, leading to irrelevant facts in responses.

We present a novel framework called PICKD for generating knowledgeable dialogues with context-relevant knowledge selection using *In-Situ* prompt tuning. The framework includes a *Knowledge Selector* which is initially trained using the *In-Situ* prompt tuning approach for retrieving relevant knowledge from structured and unstructured sources that align with the dialogue context. The *Knowledge Selector* is based on the RoBERTa [5] architecture and only the prompt tokens and classification heads are learned during training. The retrieved knowledge, along with the dialogue context, is then passed to the *Response Generator* which utilises the BART architecture for producing knowledgeable and coherent responses. In summary, our contributions are as follows:

- PICKD, a two-stage framework for Knowledge Grounded Dialogue (KGD).
- Novel *In-Situ* prompt tuning paradigm for *Knowledge Selection* in domain-specific knowledge-grounded dialogue for retrieving context-relevant knowledge.
- Extensive evaluations on multiple domain-specific knowledge-grounded dialogue datasets with ablation variants to demonstrate the strong evidential improvements of PICKD over other baselines for knowledgeable dialogue.

2 Related Work

Knowledge-grounded dialogue systems focus on generating informative responses pertinent to the dialogue context. Early works proposed using recurrent neural network-based sequence-to-sequence models for dialogue generation [9,10]. With the success of the transformer architecture, researchers explored transformer-based models for dialogue understanding and response generation [8,16]. Zhang et al. [15] proposed Dialo-GPT, a GPT-2 [7] based model trained on large-scale dialogue corpus for dialogue response generation.

However, models trained on open-domain data suffer from hallucinations and produce factually inconsistent responses. This necessitated the need for knowledge-grounded dialogue systems. Dinan et al. [1] proposed the Wizard of Wikipedia dataset wherein dialogues are grounded on Wikipedia articles. Additionally, they proposed a novel Transformer Memory Network for knowledgeable response generation. Following this, researchers explored the utilisation of reinforcement learning [17], personalisation memory [2] and knowledge internalisation [12] based methodologies for knowledgeable dialogue using unstructured knowledge sources. On the other hand, Zhu et al. [22] suggested grounding dialogue systems on structured sources such as KGs. Following this, novel frameworks [19,20] leveraging structured knowledge were proposed for response generation. Another line of work has been exploring prompting frameworks [6,18] for response generation. In these prompt-based models, the dialogue context and learnable prompt tokens are sent to language models for knowledgeable dialogue.

A major drawback of such models is the inability to utilise structured and unstructured knowledge in unison. Zhou et al. [21] proposed using a memory-based Hierarchical Recurrent Neural Network for grounding dialogues on external knowledge. Following this, Wang et al. [11] suggested RT-KGD, a novel framework utilising dialogue transition for knowledgeable dialogue generation. However, RT-KGD does not represent knowledge elements and the dialogue history in the same semantic vector space leading to an information gap between the dialogue context and the knowledge elements. In this work, we propose PICKD a novel knowledge-grounded dialogue generation framework. PICKD employs a *Knowledge Selector* for retrieving appropriate knowledge aligned with the dialogue context for response generation. The *Knowledge Selector* is trained using a novel *In-Situ* prompt tuning framework, wherein the prompt tokens are injected into the knowledge and dialogue input. The *Response Generator* module of PICKD then exploits this retrieved knowledge and the dialogue context for response generation.

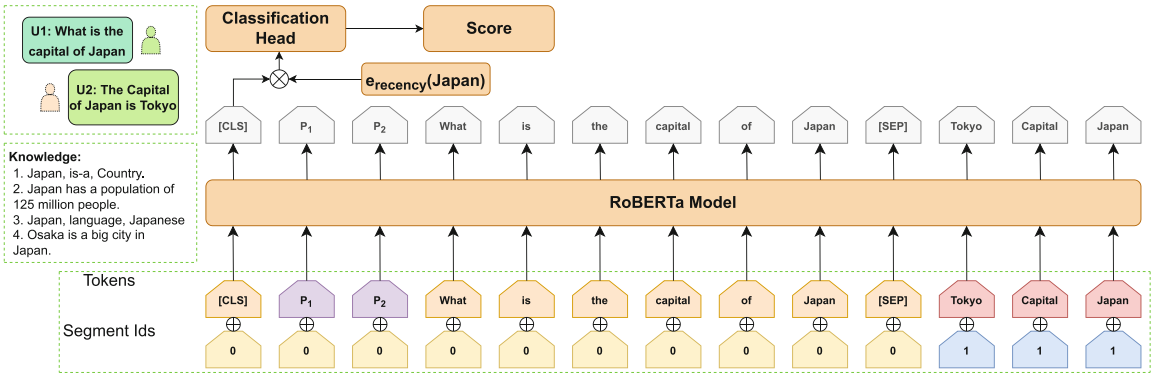

Fig. 2. Overview of the *In-Situ* prompting tuning framework which uses the RoBERTa architecture as its base model. The input token embeddings are augmented with segment token embeddings, and the output of the $[CLS]$ token embedding is then sent to a classification layer along with the recency embedding of the head entity of the knowledge element. This classification layer learns to assign higher scores to relevant knowledge. Here, $\bigoplus$ denotes addition, while $\bigotimes$ denotes the concatenation operation.

3 Methodology

We begin with formally defining the problem statement. Thereafter, we introduce the *Knowledge Selector* module, our novel *In-Situ* prompt tuning framework for knowledge selection. Finally, we describe the workings of the *Response Generator* for knowledgeable dialogue generation.

3.1 Formal Problem Definition

We are given the dialogue context $C = \{u_1, u_2, u_3, ..., u_{n-1}\}$, where u_i is an utterance from the conversation history, triples $\{(h_1, r_1, t_1), (h_2, r_2, t_2), ..., (h_{|K_k|}, r_{|K_k|}, t_{|K_k|})\}$ from a KG, where K_k is the number of triples in the KG and descriptive text set $S = \{S_1, S_2, ..., S_{S_k}\}$, where S_k is the number of articles in the set and each article comprises of multiple text sentences. The objective is to generate a coherent response u_n that is not only grammatically correct but also informative, leveraging the context and the knowledge sources available.

3.2 Contextual Prompting for Knowledge Selection

The *Knowledge Selector* in PICKD retrieves knowledge from structured and unstructured sources to be leveraged by the *Response Generator*. The *Knowledge Selector* module is trained to understand the semantic congruence between the dialogue context and knowledge elements using the novel *In-Situ* prompt tuning paradigm. The knowledge facts from structured and unstructured sources are then ranked according to their relevance by the module, and appropriate knowledge is then sent to the *Response Generator* (Fig. 2).

Prompting Architecture. The *Knowledge Selector* module of PICKD employs a pre-trained RoBERTa model as its backbone. The RoBERTa[1] model takes the

[1] https://huggingface.co/hfl/chinese-roberta-wwm-ext.

dialogue context C and the knowledge K as inputs. The knowledge K comprises a triple of an entity h from the KG, a relation r from the KG, and a tail entity t from the KG or a sentence from the paragraph description of h. As unstructured knowledge is available in paragraph form, it is split into individual sentences. To ensure the input format of knowledge from KG and paragraphs are uniform, the input t can be either the tail entity or a sentence about h from its paragraph description. The input from K to the RoBERTa is the strings of h, r and t concatenated together. The concatenation of context C and knowledge K form the initial RoBERTa input. PICKD injects prompt tokens into the input and adds segment embeddings to the resulting input tokens. This enables the capture of semantic dependencies between the context and knowledge elements. The token embeddings then pass through a RoBERTa layer, and the final representation of the $[CLS]$ token along with the recency embedding of h is sent to the classification layer for ranking. Recency embeddings are learnable vectors that capture the relevance of h in the dialogue history, similar to positional embeddings. This ensures that head entities mentioned recently have more influence during response generation. Formally, we define the setup as follows:

$$P_{prompt} = P_1, P_2, ..., P_{k_{prompt}} \tag{1}$$
$$D_{dial} = D_1 D_2 ... D_n \tag{2}$$
$$K_{know} = K_1 K_2 ... K_k \tag{3}$$
$$T_{input} = [CLS] P_{prompt} D_{dial} [SEP] K_{know} \tag{4}$$
$$h_{CLS}, h_{p_1}, ..., h_{K_k} = \text{RoBERTa}(T_{input}) \tag{5}$$
$$v = MLP(h_{CLS}; \mathbf{e}_{recency}) \tag{6}$$
$$\text{score} = Softmax(v) \tag{7}$$

where P_i, D_j and K_k denote the i^{th} prompt token, j^{th} dialogue history token and the k^{th} knowledge tokens. The input to the RoBERTa model is composed of the prompt tokens, dialogue tokens and knowledge tokens concatenated together. A $[CLS]$ token is added to the beginning of the sequence denoting start of the sequence and a $[SEP]$ token is used to separate the prompt and dialogue tokens from the knowledge tokens as shown in Eq. 4. This input is sent through a RoBERTa layer as shown in Eq. 5 wherein segment embeddings are added to the token embeddings before realising the contextual embeddings. PICKD then concatenates the RoBERTa representation of $[CLS]$ token with the head entity recency embedding and is then sent through the classification MLP layer as detailed in Eq. 6 and 7. The model is trained by minimising the cross-entropy loss. During inference, the knowledge elements are ranked following Eq. 7. The top-k elements are then sent to the *Response Generator* for response generation.

3.3 BART Fine-Tuning for Response Generation

The *Knowledge Selector* retrieves context-relevant knowledge to be utilised by the *Response Generator* for response generation. The *Response Generator* is trained to utilise the semantic information from the dialogue context and the