Multi-Document Scientific Summarization from a Knowledge Graph-Centric View

Pancheng Wang, Shasha Li†, Kunyuan Pang, Liangliang He, Dong Li, Jintao Tang†, Ting Wang†
College of Computer Science and Technology,
National University of Defense Technology, Changsha, China
{wangpancheng13, shashali, pangkunyuan10, heliangliang19, lidong1,tangjintao, tingwang}@nudt.edu.cn

Abstract

Multi-Document Scientific Summarization (MDSS) aims to produce coherent and concise summaries for clusters of topic-relevant scientific papers. This task requires precise understanding of paper content and accurate modeling of cross-paper relationships. Knowledge graphs convey compact and interpretable structured information for documents, which makes them ideal for content modeling and relationship modeling. In this paper, we present **KGSum**[1], an MDSS model centred on knowledge graphs during both the encoding and decoding process. Specifically, in the encoding process, two graph-based modules are proposed to incorporate knowledge graph information into paper encoding, while in the decoding process, we propose a two-stage decoder by first generating knowledge graph information of summary in the form of descriptive sentences, followed by generating the final summary. Empirical results show that the proposed architecture brings substantial improvements over baselines on the Multi-Xscience dataset.

1 Introduction

Nowadays, the exponential increasing publication rate of scientific papers makes it difficult and time-consuming for researchers to keep track of the latest advances. **M**ulti-**D**ocument **S**cientific **S**ummarization (MDSS) is therefore introduced to alleviate this information overload problem by generating succinct and comprehensive summary from clusters of topic-relevant scientific papers (Chen et al., 2021; Shah and Barzilay, 2021).

In MDSS, paper content modeling and cross-paper relationship modeling are two main issues. (1) Scientific papers contain complex concepts, technical terms, and abbreviations that convey important information about paper content. However, some previous works (Wang et al., 2018; Jiang et al., 2019) treat all text units equally, which inevitably ignore the salient information of some less frequent technical terms and abbreviations. (2) Furthermore, there exist intricate relationships between papers in MDSS, such as sequential, parallel, complementary and contradictory (Luu et al., 2021), which play a vital role in guiding the selection and organization of different contents. The latest work (Chen et al., 2021) attempt to capture cross-paper relationships via seq2seq model without considering any links between fine-grained text units. Failure to take into account explicit relationships between papers prevents their model from learning cross-paper relationships effectively.

To address the two aforementioned issues, we consider leveraging salient text units, namely entities, and their relations for MDSS. Scientific papers contain multiple domain-specific entities and relations between them. These entities and relations succinctly capture important information about the main content of papers. Knowledge graphs based on these scientific entities and relations can be inherently used for content modeling of scientific papers. Take Figure 1 as an example. The knowledge graph at the top left illustrates the main content of paper 1, which can be formulated as: *Paper 1 uses memory augmented networks method to solve the life-long one-shot learning task, the evaluation is based on image classification benchmark datasets*. Furthermore, knowledge graphs can effectively capture cross-paper relationships through entity interactions and information aggregation. In Figure 1, paper 1, 2 and 3 adopt the same method *memory networks* to solve different tasks. This relationship is demonstrated in the graph of gold summary by sharing the method node *memory networks*.

In this paper, we develop a Transformer-based (Vaswani et al., 2017) abstractive MDSS model, which can leverage knowledge graphs to

†Corresponding authors.
[1]https://github.com/muguruzawang/KGSum

Proceedings of the 29th International Conference on Computational Linguistics, pages 6222–6233
October 12–17, 2022.

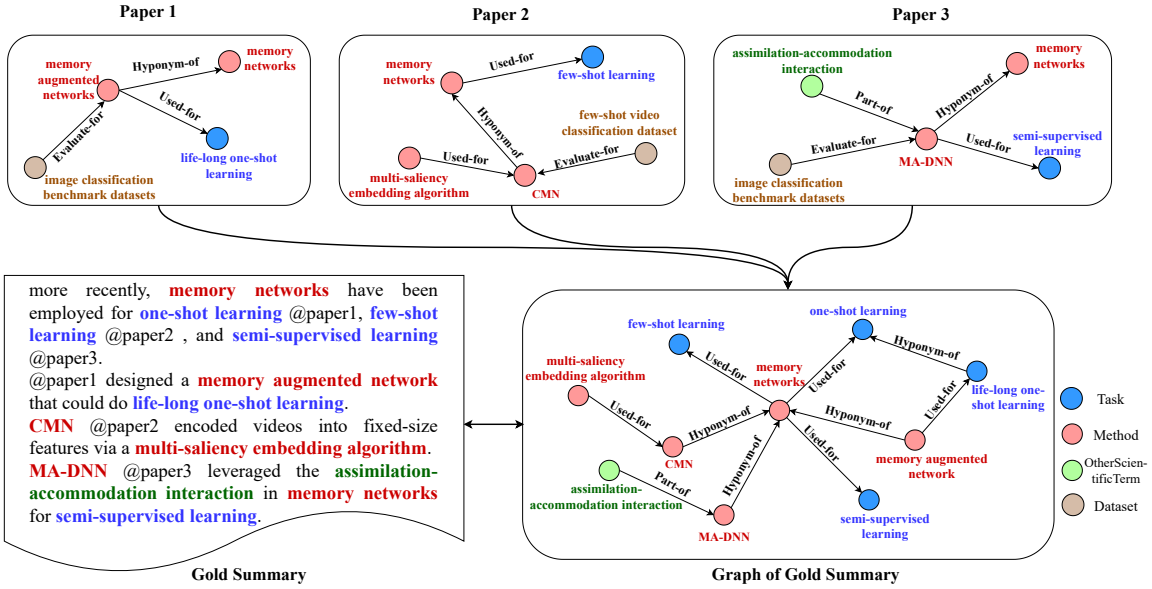


Figure 1: Knowledge graphs constructed from abstract of input scientific papers and gold summary.

guide paper representation and summary generation. Specifically, in the encoding part, we fuse the knowledge graphs of multiple input papers into a unified graph and design a graph updater to capture cross-paper relationships and global information. Besides, we build another graph based on the interaction between entities and sentences, and then apply an entity-sentence updater to enable information flow between nodes and update sentence representations.

In the decoding part, knowledge graphs are utilized to guide the summary generation process via two approaches. The first is to incorporate the graph structure into the decoder by graph attention, and the second is inspired by deliberation mechanism (Xia et al., 2017; Li et al., 2019). Concretely, we introduce a two-stage decoder to make better use of the guidance information of knowledge graphs. The first-stage decoder concentrates on generating the knowledge graph of gold summary, while the second-stage decoder generates the summary based on the output of the first stage and the input papers. Since the knowledge graph of gold summary is in the form of graph structure, we translate the graph into equivalent descriptive sentences containing corresponding entities and relations, called **KGtext**. KGtext serves as an information-intact alternative to the knowledge graph of gold summary and is generated in the first-stage decoder, which we call the KGtext generator.

We test the effectiveness of our proposed model on Multi-XScience (Lu et al., 2020), a large-scale dataset for MDSS. Experimental results show that our proposed knowledge graph-centric model achieves considerable improvement compared with the baselines, indicating that knowledge graphs can exert a positive impact on paper representation and summary generation.

The main contribution is threefold: (i) We leverage knowledge graphs to model content of scientific papers and cross-paper relationships, and propose a novel knowledge graph-centric model for MDSS. (ii) We propose a two-stage decoder that introduces KGtext as intermediate output when decoding, which plays an important guiding role in the final summary generation. (iii) Automatic and human evaluation results on the Multi-Xscience dataset show the superiority of our model.

2 Approach

2.1 Problem Formulation

We first introduce the problem formulation and used notations for MDSS. Given a set of query-focused scientific papers $\mathcal{D} = \{d_1, d_2, ..., d_N\}$, where N denotes the number of input papers. Each paper d_i consists of M_i sentences $\{s_{i,1}, s_{i,2}, ...s_{i,M_i}\}$, while each sentence $s_{i,j}$ consists of $K_{i,j}$ words $\{w_{i,j,1}, w_{i,j,2}, ..., w_{i,j,K_{i,j}}\}$. The gold summary $S = \{w_1, w_2, ...w_{N_s}\}$, N_s is the number of words in the gold summary. The target is to generate a summary $\hat{S} = \{\hat{w}_1, \hat{w}_2, ...\hat{w}_{N_{\hat{s}}}\}$ that is close enough to the gold summary S.

In our two-stage decoder framework, the gold KGtext $T = \{w_{t_1}, w_{t_2}, ..., w_{t_{\hat{N}}}\}$ is also attached

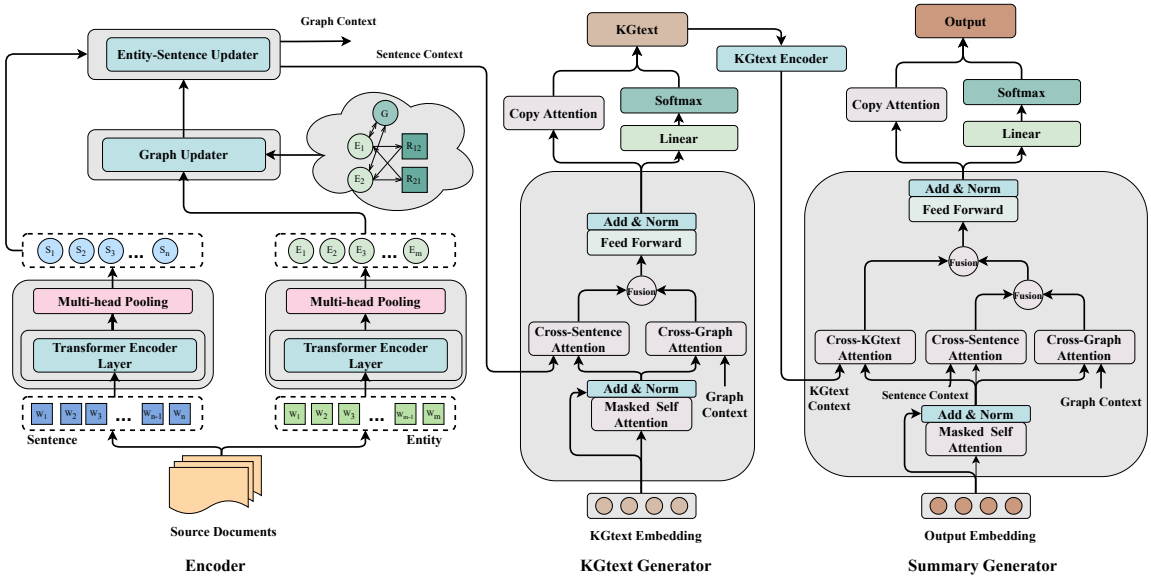


Figure 2: The overall framework of our proposed model.

as input. Hence, the probability of generating the gold summary S is

$$P(S|\mathcal{D}) = P_{\theta_{\mathcal{D}\to T}}(T|\mathcal{D}) * P_{\theta_{(\mathcal{D},T)\to S}}(S|\mathcal{D},T) \quad (1)$$

where $\theta_{\mathcal{D}\to T}$ and $\theta_{(\mathcal{D},T)\to S}$ are the parameters for the first-stage KGtext generator and the second-stage summary generator, respectively.

2.2 Graph Construction

To construct the knowledge graphs for input papers, we first employ the SciIE system DYGIE++ (Wadden et al., 2019), a well-performed science-domain information extraction system, to extract entities, relations and co-references from papers. Entities are classified into six types (*Task, Method, Metric, Material, Generic*, and *OtherScientificTerm*), and relations are classified into seven types (*Compare, Used-for, Feature-of, Hyponym-of, Evaluate-for, Part-of*, and *Conjunction*). Besides, we collapse coreferential entity clusters into a single node based on the annotation result.

After obtaining the knowledge graphs of multiple input papers, we fuse them into a unified graph. Then we follow the Levi transformation (Levi, 1942) to treat each entity and relation equally. Concretely, each labeled edge is represented as two vertices: one denoting the forward relation and another denoting the reverse relation. Formally, given an entity-relation tuple (e_1, r, e_2), we create nodes e_1, e_2, r_1 and r_2, and add directed edges $e_1 \to r_1, r_1 \to e_2$ and $e_2 \to r_2, r_2 \to e_1$. In this way, the original knowledge graph is reconstructed as an unlabeled directed graph without information loss. Besides, to guarantee the connectivity of Levi graph, we add a global vertex that connects all the entity vertices. We also add entity type nodes and connect all the entities to their corresponding types.

2.3 Model Description

Our model follows a Transformer-based encoder-decoder architecture, shown in Figure 2. The encoder includes a stack of L_1 token-level Transformer encoding layers to encode contextual information for tokens within each sentence and each entity. The Transformer encoding layer follows the Transformer architecture introduced in Vaswani et al. (2017). The encoder also includes a **Graph Updater** to learn the graph representation of the knowledge graph and an **Entity-Sentence Updater** to update entity representation and sentence representation based on their interaction. The decoder consists of a **KGtext Generator**, which produces the descriptive sentences of the graph of gold summary, and a **Summary Generator**, which produces the final summary.

2.4 Graph Updater

As shown in Figure 2, based on the output of token-level Transformer encoding layers, the graph updater is used to encode the knowledge graphs to obtain graph representations of input papers.

Node Initialization The vertices of the constructed graph correspond to entities, relations and entity types from the SciIE annotations. Entities

representations are produced using the aforementioned Transformer-based encoding method. For a given entity co-reference cluster, we first remove pronouns and stopwords and then obtain the entity representation by using the average embedding of entities in the cluster. For relation representation, since each relation is represented as both forward and backward vertices, we learn two embeddings per relation. We also randomly initialize the types embeddings and the global vertex embedding.

Contextualized Node Encoding We follow Koncel-Kedziorski et al. (2019) and use a Graph Transformer to compute the hidden representations of each node in the graph. Graph Transformer encodes each vertex v_i using the multi-head self-attention mechanism similar to Vaswani et al. (2017), where each vertex representation $\mathbf{v}_i$ is contextualized by attending over the other vertices to which v_i is connected in the graph.

$$\hat{\mathbf{v}}_i = \mathbf{v}_i + \|_{n=1}^{N} \sum_{v_j \in \mathcal{N}_i} \alpha_{i,j}^n \mathbf{W}_V^n \mathbf{v}_j \quad (2)$$

$$\alpha_{i,j}^n = \text{softmax}((\mathbf{W}_K^n \mathbf{v}_j)^T (\mathbf{W}_Q^n \mathbf{v}_i)) \quad (3)$$

where $\|_{n=1}^{N}$ means the concatenation of N heads. $\mathcal{N}_i$ denotes the neighbors of v_i, and $\mathbf{W}_Q^n$, $\mathbf{W}_K^n$, and $\mathbf{W}_V^n$ are trainable parameters of query, key and value of head n, respectively.

2.5 Entity-Sentence Updater

After getting the contextualized node embeddings for the knowledge graph, we construct an entity-sentence heterogeneous graph to update sentence representations based on the interaction between entities and sentences. The entity-sentence graph is denoted as $G = \{V, E\}$, where V stands for nodes set and E stands for edges set. In the graph G, V includes entity nodes V_e and sentence nodes V_s, and E is a real-value edge weight matrix, where $e_{i,j} \neq 0$ indicates the j-th sentence contains the i-th entity.

We apply the same Graph Transformer module as the graph updater. It takes as input the entities representations from the graph updater and the sentence representations from the Transformer encoding layer, then learns the representations of nodes based on the information flow through the graph G.

2.6 KGtext Generator

In the decoding stage, we also adopt the knowledge graph-centric view and introduce the KGtext generator before the final summary generator. Here, KGtext is defined as descriptive sentences containing entities and relations translated from the knowledge graph of gold summary. An example of KGtext is shown in Figure 3.

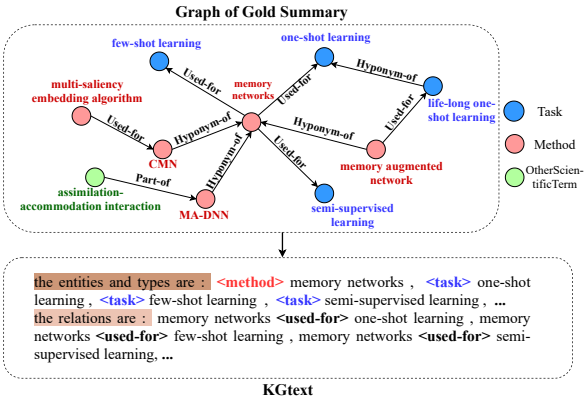


Figure 3: An example of graph of gold summary and the translated KGtext.

KGtext Construction To construct KGtext, we first use DYGIE++ (Wadden et al., 2019) to extract entities and relations from the human-written gold summary of the training set. Then we fill the KGtext with the prefix *The entities and types are:* followed by each entity type and entity pair like *<TYPE> ENT*. We also add another prompt *the relations are:* to introduce the relations, in the form of *ENT_1 <REL> ENT_2*.

KGtext serves as an information-intact alternative to the knowledge graph of gold summary, which is generated by the KGtext generator and can provide knowledge graphs information for the final summary generation.

Decoding Since the knowledge graph of gold summary is obtained by synthesizing and simplifying the knowledge graphs of input papers via the interaction of nodes, the graph structure plays an important role in KGtext generation. Hence during decoding, we leverage source token representations as well as graph representations during KGtext decoding process.

We apply a stack of L_2 Transformer decoding layers as the decoder. The cross-attention sub-layer of each decoding layer computes two multi-head attention to capture both textual and graph context. Let $\tilde{g}_i^l$ denotes the i-th token output representation by the l-th self-attention sub-layer. For the textual context, we use $\tilde{g}_i^l$ as query and token representations $\mathbf{H_W}$ from entity-sentence updater as keys

and values.

$$c_{i,w}^{l} = \mathrm{MHAtt}(\tilde{g}_i^l, \mathbf{H_W}, \mathbf{H_W}) \quad (4)$$

where MHAtt denotes the multi-head attention module proposed in Vaswani et al. (2017).

For the graph context, we use $\tilde{g}_i^l$ as query and entity nodes representations $\mathbf{H_E}$ from entity-sentence updater as keys and values. Considering that different entities of the input have different importance, we apply the unsupervised phrase scoring algorithm RAKE (Rose et al., 2010) to score the salience of entities, and incorporate entity salience into graph context computation. Given the RAKE scores $S = \{s_j\}$ for entity nodes representations $\mathbf{H_E}$, we modify MHAtt module by multiplying S with the attention weights.

$$c_{i,g}^{l} = \mathrm{MHAtt\_Mod}(\tilde{g}_i^l, \mathbf{H_E}, \mathbf{H_E}, S) \quad (5)$$

where MHAtt_Mod denotes the modified MHAtt module. And the modified attention weight α_t^n of head n is calculated as

$$\alpha_i^n = \frac{(\mathrm{W}_K^n \mathbf{H_E})^T (\mathrm{W}_Q^n \tilde{g}_i^l)}{\sqrt{d_{head}}} * S \quad (6)$$

where W_K^n and W_Q^n are parameter weights, d_{head} denotes the dimension of each attention head.

We then add a fusion gate to merge both the textual context and the graph context.

$$c_i^l = z * c_{i,w}^l + (1 - z) * c_{i,g}^l \quad (7)$$

$$z = \mathrm{sigmoid}([c_{i,w}^l; c_{i,g}^l]\mathrm{W}_f + b_f) \quad (8)$$

where W_f and b_f are the linear transformation parameter. The feed-forward network is used to further transform the output.

$$g_i^l = \mathrm{LayerNorm}(c_i^l + \mathrm{FFN}(c_i^l)) \quad (9)$$

The generation distribution p_t^g over the target vocabulary is calculated by feeding the output $g_t^{L_2}$ to a softmax layer.

$$p_i^g = \mathrm{softmax}(g_i^{L_2}\mathrm{W}_g + b_g) \quad (10)$$

where W_g and b_g are learnable linear transformation parameter.

Furthermore, we also employ the copy mechanism (See et al., 2017) to alleviate the out-of-vocabulary (OOV) problem. The final generation distribution p_i^t is the "mixture" of both p_i^g and the copy probability over source words p_i^c.

The loss is the negative log likelihood of the gold KGtext w_{t_i}:

$$L_T = -\frac{1}{\hat{N}} \sum_{i=1}^{\hat{N}} \log p_i^t(w_{t_i}) \quad (11)$$

2.7 Summary Generator

The final summary generator has a similar decoding architecture to the KGtext generator, but differs in that the summary generator utilizes the generated KGtext to guide summary generation.

Given the KGtext generative distribution $\{p_i^t\}$, we obtain the decoding sequence of KGtext $\hat{T}$ by greedy search during training. Then we add an encoder similar to the aforementioned sentence encoder to get the KGtext representations $\mathbf{H_T}$. Besides attending to textual and graph context, we use the same multi-head attention as equation (4) to compute KGtext context $\hat{c}_{i,t}^l$ to capture KGtext influence.

Together with the textual context $\hat{c}_{i,w}^l$ and the graph context $\hat{c}_{i,g}^l$, we apply a hierarchical fusion mechanism to combine the three contexts, by first merging the textual context and the graph context, and then the KGtext context.

$$\hat{c}_i^l = z_1 * \hat{c}_{i,w}^l + (1 - z_1) * \hat{c}_{i,g}^l \quad (12)$$

$$z_1 = \mathrm{sigmoid}([\hat{c}_{i,w}^l; \hat{c}_{i,g}^l]\mathrm{W}_{1,f} + b_{1,f}) \quad (13)$$

$$c_i^l = z_2 * \hat{c}_i^l + (1 - z_2) * \hat{c}_{i,t}^l \quad (14)$$

$$z_2 = \mathrm{sigmoid}([\hat{c}_i^l; \hat{c}_{i,t}^l]\mathrm{W}_{2,f} + b_{2,f}) \quad (15)$$

where $\mathrm{W}_{1,f}$, $b_{1,f}$, $\mathrm{W}_{2,f}$ and $b_{2,f}$ are the linear transformation parameter.

Given the final summary generation distribution p_i^s, the loss is the negative log likelihood of the gold summary w_i:

$$L_S = -\frac{1}{N_s} \sum_{i=1}^{N_s} \log p_i^s(w_i) \quad (16)$$

2.8 Training Strategy

We train the KGtext generator and the summary generator in a unified architecture in an end-to-end manner. Furthermore, in practice, we find the KGtext generated from greedy search has a strong influence on the summary generation. The low-quality KGtext greatly impairs the performance of the model. Hence, we train another auxiliary decoder on top of $P_{\theta_{\mathcal{D}\to\mathrm{S}}}(S|\mathcal{D})$, which uses a one-stage decoder without generating KGtext. It has the same architecture as the summary generator except for the cross-attention on KGtext.

Given the final summary generation distribution of the auxiliary decoder $\tilde{p}_i^s$, the loss is the negative log likelihood of the gold summary w_i:

$$L_A = -\frac{1}{N_s} \sum_{i=1}^{N_s} \log \tilde{p}_i^s(w_i) \quad (17)$$

The final loss function is:

$$\mathcal{L} = \mathcal{L}_S + \lambda \mathcal{L}_T + \eta \mathcal{L}_A \quad (18)$$

where λ and η are both hyper parameters. In this way, we can reduce the effect of some low-quality generated KGtext and improve the stability of our model.

3 Experiments

3.1 Dataset

We conduct experiments on the recently released Multi-Xscience dataset (Lu et al., 2020), which is the first large-scale and open MDSS dataset. It contains 30,369 instances for training, 5,066 for validation and 5,093 for test. On average, each source paper cluster contains 4.42 papers and 778.08 words, and each gold summary contains 116.44 words.

3.2 Implementation Details

We set our model parameters based on preliminary experiments on the validation set. We prune the vocabulary to 50K. The number of encoding layer L_1 and the number of decoding layer L_2 are both 6. We set the dimension of word embeddings and hidden size to 256, feed-forward size to 1,024. We set 8 heads for multi-head attention. For the Graph Transformer of the graph updater and the entity-sentence updater, we set the number of iterations to 3. We set dropout rate to 0.1 and label smoothing (Szegedy et al., 2016) factor to 0.1. We use Adam optimizer with learning rate $\alpha = 0.02$, $\beta_1 = 0.9$, $\beta_2 = 0.998$; we also apply learning rate warmup over the first 8000 steps, and decay as in Vaswani et al. (2017). The batch size is set to 8. λ and η are both set to 1.0. The model is trained on 1 GPU (NVIDIA Tesla V100, 32G) for 100,000 steps. We select the top-3 best checkpoints based on performance on the validation set and report averaged results on the test set.

For KGtext decoding, we use greedy search with the minimal generation length 100, while for summary decoding, we use beam search with beam size 5 and the minimal generation length is 110, consistent with Lu et al. (2020). The length penalty factor is set to 0.4.

3.3 Metrics and Baselines

We use ROUGE F_1 (Lin, 2004) to evaluate the summarization quality. Following previous work, we report unigram and bigram overlap (ROUGE-1 and ROUGE-2) as a means of assessing informativeness and the longest common subsequence (ROUGE-L) as a means of assessing fluency.

We compare our model with several typical extractive and abstractive summarization models. Due to space limitations, we put the introduction of these models in appendix A.

Model	R-1	R-2	R-L
Extractive			
LexRank (Erkan and Radev, 2004)	30.19	5.53	26.19
TextRank (Mihalcea and Tarau, 2004)	31.51	5.83	26.58
HeterSumGraph* (Wang et al., 2020)	31.36	5.82	27.41
Ext-Oracle	38.45	9.93	27.11
Abstractive			
GraphSum* (Li et al., 2020)	29.58	5.54	26.52
Hiersumm (Liu and Lapata, 2019a)	30.02	5.04	27.6
HiMAP (Fabbri et al., 2019)	31.66	5.91	28.43
BertABS (Liu and Lapata, 2019b)	31.56	5.02	28.05
BART (Lewis et al., 2020)	32.83	6.36	26.61
SciBertABS (Lu et al., 2020)	32.12	5.59	29.01
MGSum* (Jin et al., 2020)	33.11	6.75	29.43
Pointer-Generator (See et al., 2017)	34.11	6.76	30.63
KGSum	**35.77**	**7.51**	**31.43**

Table 1: ROUGE F1 evaluation results on the test set of Multi-Xscience. The results marked with * are obtained by running the released code using the same beam size and decoding length. Other results without * are from Lu et al. (2020).

3.4 Automatic Evaluation

Table 1 summarizes the evaluation results on the Multi-Xscience dataset.

As can be seen, abstractive models generally outperform extractive models, especially on ROUGE-L, showing that abstractive models can generate more fluent summaries. Among the abstractive baselines, Pointer-Generator (See et al., 2017) surprisingly outperforms other newer models. We partially attribute this result to Pointer-Generator designing an additional coverage mechanism (Tu et al., 2016) to effectively reduce redundancy. This result also illustrates that MDSS is challenging and requires domain-specific solutions for paper content representation and cross-paper relationship modeling.

The last block reports the result of our model KGSum. KGSum outperforms any other models, achieving scores of 35.77, 7.51, and 31.43 on the three ROUGE metrics. Our model surpasses other models by a remarkable large margin (at least improving 1.66%, 0.75%, and 0.80%). The result

Model	Overall	Inf	Fluency	Succ
GraphSum	-1.42*	-1.47*	-1.08*	-1.23*
MGSum	-0.38*	0.60	-0.20*	-0.55*
Pointer-Generator	0.62*	0.31*	0.17*	0.60*
KGSum	**1.30**	**0.68**	**1.17**	**1.22**

Table 2: Human evaluation of system summaries on Multi-Xscience test set. Inf stands for *informativeness* and Succ stands for *succinctness*. The larger rating denotes better summary quality. * indicates the ratings of the corresponding model are significantly (by Welch's t-test with $p < 0.05$) outperformed by our model. The inter-annotator agreement score (Fleiss Kappa) is 0.63, which indicates substantial agreement between annotators.

Model	R-1	R-2	R-L
KGSum	**35.77**	**7.51**	**31.43**
- KGG	35.34	7.28	30.91
- KGG - RAKE	35.17	7.18	30.75
- KGG - RAKE - GU	34.97	7.08	30.63
- KGG - RAKE - GU - ESU	34.79	6.90	30.36

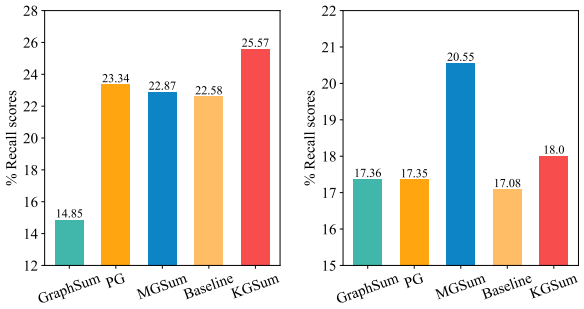
Table 3: Ablation studies on Multi-Xscience test set. We remove various modules and explore their influence on our model. '-' means the removal operation from KGSum. The last row (-KGG-RAKE-GU-ESU) is the clean baseline without any module we propose.

demonstrates that our model can generate more informative and more coherent summaries, indicating the effectiveness of our proposed knowledge graph-centric encoder and decoder framework.

3.5 Human Evaluation

We further access the linguistic quality of generated summaries by human evaluation. We randomly select 30 test instances from the Multi-Xscience test set, and invite three graduate students as annotators to evaluate the outputs of different models independently. Annotators assess the overall quality of summaries by ranking them considering the following criteria:: (1) *Informativeness*: does the summary convey important facts of the input papers? (2) *Fluency*: is the summary fluent and grammatical? (3) *Succinctness*: whether the summary contains repeated information? Annotators are asked to rank all systems from 1 (best) to 4 (worst). All systems get score 2, 1, -1, -2 for ranking 1, 2, 3, 4 respectively. The rating of each system is computed by averaging the scores on all test instances.

The result is shown in Table 2. The overall rating and the ratings for the above three aspects are reported. We can see that KGSum performs much



(a) Recall of EW (b) Recall of CW

Figure 4: Recall of EW and CW for different models on Multi-Xscience test set. PG stands for Pointer-Generator, Baseline is our Transformer-based model without any module we propose.

better than other models. The overall rating of KGSum achieves 1.2, which is much higher than 0.62, -0.38, and -1.42 of Pointer-Generator, MGSum, and GraphSum. The results on informativeness indicate our model can effectively capture the salient information of papers and generate more informative summaries. The results on fluency and succinctness indicate that KGSum is able to generate more fluent and concise summaries. Furthermore, Pointer-Generator achieves a much higher score on succinctness than MGSum, which further proves that Pointer-Generator generates less redundant summaries and thus has better performance.

3.6 Model Analysis

For a thorough understanding of KGSum, we conduct several experiments on Multi-Xscience test set.

Ablation Study Firstly, we perform an ablation study to validate the effectiveness of individual components. Here, **KGG** stands for KGtext generator, **RAKE** refers to the RAKE algorithm that measures entity salience, **GU** stands for graph updater, **ESU** stands for entity-sentence updater. We remove KGG, RAKE, GU, ESU one by one in order from decoder to encoder. The result is illustrated in Table 3. We find that the GU and ESU module in the encoder can effectively encode knowledge graph information and utilize knowledge graphs to enable better information flowing between text nodes, which is conducive to content modeling and relationship modeling. Using RAKE to measure entity salience also benefits a lot for graph context computation when decoding. Further, the KGG module also brings significant improvement, indi-

KGtext Variants	R-1	R-2	R-L
Ent	35.61	7.43	31.24
Ent+Type	35.67	7.42	31.29
Ent+Type+Rel	**35.77**	**7.51**	**31.43**

Table 4: Analysis of the impact of different KGtext contents on summarization.

cating our proposed two-stage decoder with KGtext generator is effective in generating summary under the guidance of knowledge graphs.

Recall of Entity Words In order to intuitively demonstrate the impact of knowledge graphs, we investigate the recall of gold summary entities in the generated summary. The exact match of entities is difficult because entities have different mentions. Therefore, we count recall of entity words instead. We classify the words in papers into two categories: **Entity Words** (EW) and **Context Words** (CW). EW are defined as words in papers that are recognized as entities by SciIE tools, while CW are words other than EW. We exclude stopwords when calculating EW and CW, because stopwords have no practical meaning. Then, we define *Recall of EW* as:

$$Recall_{EW} = \frac{\sum_{S \in \{Ref\}} \sum_{N_{EW} \in S} Count_{match}(N_{EW})}{\sum_{S \in \{Ref\}} \sum_{N_{EW} \in S} Count(N_{EW})} \quad (19)$$

where $\{Ref\}$ denotes the gold summaries, $Count_{match}(N_{EW})$ denotes the number of overlapped EW in the gold summaries and the generated summaries. $Count(N_{EW})$ denotes the number of EW in the gold summaries. *Recall of CW* is defined in a similar manner.

The results are shown in Figure 4. We find that KGSum achieves the highest recall of EW, compared with the baseline model and other models. The result proves that our model focuses on more entity information under the guidance of knowledge graphs. Conversely, in Figure 4b, MGSum achieves the highest recall of CW, but ROUGE-1/2/L scores of MGSum are only 33.11/6.75/29.43, falling behind KGSum. The result indicates that recall of CW has limited effect on model performance, which is in line with our intuition since CW do not contain important semantic information.

Influence of KGtext Contents We also conduct experiments to analyze the impact of different KGtext contents on MDSS. We consider the following three variants: (1) only entities (Ent), (2) entities + types (Ent+Type), (3) entities + types + relations (Ent+Type+Rel), to construct the KGtext using the same strategy in section 2.6. Result in Table 4 demonstrates MDSS can benefit from different components of knowledge graph, including entities, types and relations.

4 Related Work

Early MDSS works are mainly based on artificially constructed small-scale datasets, using unsupervised extractive methods to extract sentences from multiple papers. Mohammad et al. (2009) take citation texts, paper abstracts and full paper texts as input for survey generation. They conduct the experiment with just two instances. Jha et al. (2015a) construct 15 instances and combine a content model with a discourse model to generate coherent scientific summarizations. Hoang and Kan (2010) construct 20 instances, each with an annotated topic hierarchy tree, to generate summarization for multiple scientific papers. Similar works also exist in (Jha et al., 2015b; Hu and Wan, 2014; Yang et al., 2017). These unsupervised works are crude in both content modeling and relationship modeling and fail to generate high-quality summaries.

Some subsequent efforts apply deep learning-based methods to MDSS using large-scale datasets (Wang et al., 2018; Jiang et al., 2019; Chen et al., 2021). Wang et al. (2018) build a dataset with 8080 instances and construct a heterogeneous bibliography graph, and then utilize a CNN and RNN-based model for extractive summarization. Jiang et al. (2019) collect 390,000 instances, and use a hierarchical encoder and a two-step decoder to generate summary in an abstractive manner for the first time. Chen et al. (2021) collect two large-scale datasets with 136,655 and 80,927 instances, respectively. They apply a Transformer-based model to capture the relevance between papers for abstractive summarization. However, all the above works neglect salient semantic units to capture semantic information and relationships between papers. In this paper, based on Mutli-Xscience (Lu et al., 2020), we use knowledge graph information to model content and relationships between papers.

5 Conclusion

In this work, we propose a knowledge graph-centric Transformer-based model for MDSS. Our model is able to incorporate knowledge graph information

into the paper encoding process with a graph updater and an entity-sentence updater, and introduce a two-stage decoder including a KGtext generator and a summary generator to guide the summary decoding with knowledge graph information. Experiments show that the proposed model significantly outperforms all strong baselines and achieves the best result on the Multi-Xscience dataset.

In the future, we will explore other more intuitive and effective methods to incorporate graph information in both the encoding and decoding phase of summary generation.

Acknowledgements

This work was supported by the National Key Research and Development Project of China (No. 2021ZD0110700) and Hunan Provincial Natural Science Foundation (Grant Nos. 2022JJ30668). The authors would like to thank the anonynous reviewers for their valuable comments and suggestions to improve this paper.

References

Xiuying Chen, Hind Alamro, Mingzhe Li, Shen Gao, Xiangliang Zhang, Dongyan Zhao, and Rui Yan. 2021. Capturing relations between scientific papers: An abstractive model for related work section generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 6068–6077.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186.

Günes Erkan and Dragomir R Radev. 2004. Lexrank: Graph-based lexical centrality as salience in text summarization. *Journal of artificial intelligence research*, 22:457–479.

Alexander Richard Fabbri, Irene Li, Tianwei She, Suyi Li, and Dragomir Radev. 2019. Multi-news: A large-scale multi-document summarization dataset and abstractive hierarchical model. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1074–1084.

Cong Duy Vu Hoang and Min-Yen Kan. 2010. Towards automated related work summarization. In *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*, pages 427–435. Association for Computational Linguistics.

Yue Hu and Xiaojun Wan. 2014. Automatic generation of related work sections in scientific papers: an optimization approach. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1624–1633.

Rahul Jha, Reed Coke, and Dragomir Radev. 2015a. Surveyor: A system for generating coherent survey articles for scientific topics. In *Twenty-Ninth AAAI conference on artificial intelligence*.

Rahul Jha, Catherine Finegan-Dollak, Ben King, Reed Coke, and Dragomir Radev. 2015b. Content models for survey generation: a factoid-based evaluation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, pages 441–450.

Xiao-Jian Jiang, Xian-Ling Mao, Bo-Si Feng, Xiaochi Wei, Bin-Bin Bian, and Heyan Huang. 2019. Hsds: An abstractive model for automatic survey generation. In *International Conference on Database Systems for Advanced Applications*, pages 70–86. Springer.

Hanqi Jin, Tianming Wang, and Xiaojun Wan. 2020. Multi-granularity interaction network for extractive and abstractive multi-document summarization. In *Proceedings of the 58th annual meeting of the association for computational linguistics*, pages 6244–6254.

Rik Koncel-Kedziorski, Dhanush Bekal, Yi Luan, Mirella Lapata, and Hannaneh Hajishirzi. 2019. Text generation from knowledge graphs with graph transformers. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2284–2293.

Friedrich Wilhelm Levi. 1942. Finite geometrical systems.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880.

Wei Li, Xinyan Xiao, Jiachen Liu, Hua Wu, Haifeng Wang, and Junping Du. 2020. Leveraging graph to improve abstractive multi-document summarization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6232–6243.

Zekang Li, Cheng Niu, Fandong Meng, Yang Feng, Qian Li, and Jie Zhou. 2019. Incremental transformer with deliberation decoder for document grounded conversations. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 12–21.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81.

Yang Liu and Mirella Lapata. 2019a. Hierarchical transformers for multi-document summarization. *arXiv preprint arXiv:1905.13164*.

Yang Liu and Mirella Lapata. 2019b. Text summarization with pretrained encoders. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3730–3740.

Yao Lu, Yue Dong, and Laurent Charlin. 2020. Multi-xscience: A large-scale dataset for extreme multi-document summarization of scientific articles. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 8068–8074.

Kelvin Luu, Xinyi Wu, Rik Koncel-Kedziorski, Kyle Lo, Isabel Cachola, and Noah A Smith. 2021. Explaining relationships between scientific documents. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 2130–2144.

Rada Mihalcea and Paul Tarau. 2004. Textrank: Bringing order into text. In *Proceedings of the 2004 conference on empirical methods in natural language processing*, pages 404–411.

Saif Mohammad, Bonnie Dorr, Melissa Egan, Ahmed Hassan, Pradeep Muthukrishnan, Vahed Qazvinian, Dragomir Radev, and David Zajic. 2009. Using citations to generate surveys of scientific paradigms. In *Proceedings of human language technologies: The 2009 annual conference of the North American chapter of the association for computational linguistics*, pages 584–592.

Stuart Rose, Dave Engel, Nick Cramer, and Wendy Cowley. 2010. Automatic keyword extraction from individual documents. *Text mining: applications and theory*, 1:1–20.

Abigail See, Peter J Liu, and Christopher D Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1073–1083.

Darsh J Shah and Regina Barzilay. 2021. Generating related work. *arXiv preprint arXiv:2104.08668*.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. 2016. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826.

Zhaopeng Tu, Zhengdong Lu, Yang Liu, Xiaohua Liu, and Hang Li. 2016. Modeling coverage for neural machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 76–85.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

David Wadden, Ulme Wennberg, Yi Luan, and Hannaneh Hajishirzi. 2019. Entity, relation, and event extraction with contextualized span representations. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5784–5789.

Danqing Wang, Pengfei Liu, Yining Zheng, Xipeng Qiu, and Xuan-Jing Huang. 2020. Heterogeneous graph neural networks for extractive document summarization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6209–6219.

Yongzhen Wang, Xiaozhong Liu, and Zheng Gao. 2018. Neural related work summarization with a joint context-driven attention mechanism. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1776–1786.

Yingce Xia, Fei Tian, Lijun Wu, Jianxin Lin, Tao Qin, Nenghai Yu, and Tie-Yan Liu. 2017. Deliberation networks: Sequence generation beyond one-pass decoding. *Advances in neural information processing systems*, 30.

Shansong Yang, Weiming Lu, Dezhi Yang, Xi Li, Chao Wu, and Baogang Wei. 2017. Keyphraseds: Automatic generation of survey by exploiting keyphrase information. *Neurocomputing*, 224:58–70.

Jingqing Zhang, Yao Zhao, Mohammad Saleh, and Peter Liu. 2020. Pegasus: Pre-training with extracted gap-sentences for abstractive summarization. In *International Conference on Machine Learning*, pages 11328–11339. PMLR.

A Baselines

LexRank (Erkan and Radev, 2004) and **TextRank** (Mihalcea and Tarau, 2004) are two unsupervised graph based extractive summarization models. **HeterSumGraph** (Wang et al.,

2020) is a heterogeneous graph-based extractive model with semantic nodes of different granularity. **HiMAP** (Fabbri et al., 2019) expands the pointer-generator network (See et al., 2017) into a hierarchical network and integrates an MMR module. **HierSumm** (Liu and Lapata, 2019a) is a Transformer based model with an attention mechanism to share information cross-document for abstractive multi-document summarization. **MGSum** (Jin et al., 2020) is a multi-granularity interaction network for abstractive multi-document summarization. We also consider evaluating on single document summarization models by concatenating multiple papers into a long sequence. **GraphSum** (Li et al., 2020) is a neural multi-document summarization model that leverages well-known graphs to produce abstractive summaries. We use TF-IDF graph as the input graph. **PEGASUS** (Zhang et al., 2020) is a sequence-to-sequence model with gap-sentences generation as a pre-training objective tailored for abstractive summarization. **Pointer-Generator** (See et al., 2017) is an RNN based model with an attention mechanism and allows the system to copy words from the source via pointing for abstractive summarization. **BertABS** (Liu and Lapata, 2019b) uses a pretrained BERT (Devlin et al., 2019) as the encoder for abstractive summarization. We also report the performance of BertABS with an encoder (**SciBertABS**) pretrained on scientific articles. **BART** (Lewis et al., 2020) is a pretrained text generation model.

B Case Study

In Figure 5, we present several example summaries to show the generating quality of different models. In the figure, there are five blocks, which are the reference summary (**Reference Summary**), the system summaries generated by **Pointer-Generator**, **GraphSum**, **MGSum** and our model **KGSum**. We highlight the salient contents from system summaries that can be find in the reference summary. We could find our model KGSum possesses the most salient contents and the highest overlap with the reference summary. Pointer-Generator and MGSum have the same amount of salient contents, but MGSum contains more repeated information, leading to worse performance. GraphSum contains the least salient contents, resulting in the worst performance.

Reference Summary:
most recent approaches rely on convolutional neural networks (cnns). @cite , the work in @cite is extended by adding a cnn to describe the posterior density of an object pose. a combination of using a cnn for object segmentation and geometry-based pose estimation is proposed in @cite . posecnn @cite uses a similar two-stage network, in which the first stage extracts feature maps from rgb input and the second stage uses the generated maps for object segmentation, 3d translation estimation and 3d rotation regression in quaternion format. depth data and icp are used for geometry refinement. @cite propose a three-stage, instance-aware approach for 6d object pose estimation. an instance segmentation network is first applied, followed by an encoder-decoder network which estimates the 3d object coordinates for each segment. the 6d pose is recovered with a geometric pose optimization step similar to @cite .

Pointer-Generator:
our work is also related to the work in @cite , where the authors use a convolutional neural networks (cnns) to predict the 6d pose of specific objects in a single rgb - d image . @cite proposed a new loss function for 6d pose estimation of object pose estimation . @cite , the authors propose a method to directly regress a pose vector from raw point cloud segments using a quaternion neural network . they then applied a quaternion representation to a quaternion model , which can be used to learn the pose of the object pose . in contrast to our work , our method is able to learn pose estimation from a single image

GraphSum:
3d object pose estimation has been a long-standing research topic in computer vision and computer vision @cite @cite @cite . in recent years, there has been significant interest in object reconstruction from a single image @cite @cite or 3d object @cite @cite , and many of them have been trying to recover from single images @cite . however, most of these methods rely on the availability of a large number of training images. in contrast, our approach does not require a large amount of labeled training data. in addition, our approach is able to estimate the 3d pose of a single object.

MGSum:
posecnn @cite is a flexible approach for 6d object pose estimation. it uses a cnn to predict 6d pose from a single image. posecnn @cite predicts 6d pose by regressing the 3d rotation of a 3d object by regressing a quaternion representation. posecnn @cite uses a quaternion network to predict the 6d pose using a quaternion network, and learns a quaternion representation of the object representation. posecnn @cite proposes a "learns approach for estimating 6d object pixels in an image using quaternion representation. posecnn and posecnn @cite propose an approach approach to regressing the object pixels from a particular object representation. however, these methods require a large amount of training data, which is difficult to obtain. in contrast, our method does not rely on a large number of objects in the scene.

KGSum:
6d pose estimation has been widely used in computer vision @cite @cite . in @cite , the authors propose a cnn for 6d object pose estimation. the first step is to use a quaternion to handle the 6d pose of an object in a single image. the second stage is based on the idea of using a convolutional neural networks to learn a 3d rotation of the object and the appearance of a 6d pose. the proposed method is also used for object segmentation @cite and pose estimation, which is used to refine the geometry of object proposals. @cite used an encoder-decoder network to handle the 6d pose of known objects in a rgb-d image by regressing the 3d rotation of an object with a 3d representation.

Figure 5: Case study from the Multi-Xscience test set. We first highlight the salient contents in the reference summary in different colors. Then the overlapped salient contents of system summaries are annotated in the same colors.