Exploring Key Point Analysis with
Pairwise Generation and Graph Partitioning

Xiao Li$^\diamond$, Yong Jiang†*, Shen Huang†, Pengjun Xie†
Gong Cheng$^{\diamond*}$, Fei Huang†

$^\diamond$State Key Laboratory for Novel Software Technology, Nanjing University
†Institute for Intelligent Computing, Alibaba Group
`xiaoli.nju@smail.nju.edu.cn, gcheng@nju.edu.cn`
`{yongjiang.jy,pangda,chengchen.xpj,f.huang}@alibaba-inc.com`

Abstract

Key Point Analysis (KPA), the summarization of multiple arguments into a concise collection of key points, continues to be a significant and unresolved issue within the field of argument mining. Existing models adapt a two-stage pipeline of clustering arguments or generating key points for argument clusters. This approach rely on semantic similarity instead of measuring the existence of shared key points among arguments. Additionally, it only models the intra-cluster relationship among arguments, disregarding the inter-cluster relationship between arguments that do not share key points. To address these limitations, we propose a novel approach for KPA with pairwise generation and graph partitioning. Our objective is to train a generative model that can simultaneously provide a score indicating the presence of shared key point between a pair of arguments and generate the shared key point. Subsequently, to map generated redundant key points to a concise set of key points, we proceed to construct an arguments graph by considering the arguments as vertices, the generated key points as edges, and the scores as edge weights. We then propose a graph partitioning algorithm to partition all arguments sharing the same key points to the same subgraph. Notably, our experimental findings demonstrate that our proposed model surpasses previous models when evaluated on both the ArgKP and QAM datasets.

1 Introduction

Analyzing a large number of arguments and making decisions is a long standing issue present in real life scenarios such as product review analysis (Bar-Haim et al., 2021; Cattan et al., 2023), opinion surveys (Bar-Haim et al., 2020a), etc. However, manually reading through a vast amount of arguments is not a feasible solution. In this context,
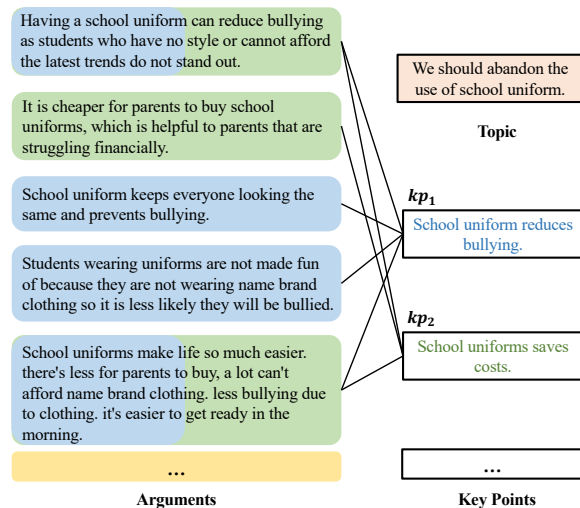

Figure 1: An example in the ArgKP dataset. Only a subset of arguments and key points are shown due to space limitation. Our goal is to generate key points (right side) within a specific topic based on the provided arguments (left side). A single argument may correspond to multiple key points.

key point analysis (KPA, Bar-Haim et al., 2020a) has been proposed. KPA involves distilling a multitude of arguments into a concise set of key points, followed by a quantitative evaluation, typically by calculating the frequency of each key point related to arguments. An example is shown in Figure 1.

Existing Methods and Limitations. The generation of key points from arguments represents the most pivotal and challenging aspect of the KPA task. Initially, a straightforward approach (Cattan et al., 2023) involves selecting a few short arguments as key points and then determining whether other arguments match these key points. Subsequently, advanced model (Li et al., 2023) divides the KPA task into two stages. Firstly, a semantic feature based clustering model is utilized to cluster arguments into multiple clusters. Then, a generation model is employed to generate correspond-

---

ing key points for arguments in each cluster by concatenating the arguments. However, these existing methods possess certain limitations. Cattan et al., 2023 posits that the mere presence of appropriate key points within arguments is insufficient. Li et al., 2023's approach is based on clustering arguments using their semantic features and then positing that arguments within a cluster share the same key point. However, this approach does not fully align with the objectives of the KPA task. It is possible for arguments that are not semantically similar to still share a key point, as an argument can correspond to more than one key point. Moreover, Li et al., 2023's approach only models the relationships between arguments that share a key point (i.e., intra-cluster arguments) and neglects the relationships between arguments that do not share a key point (i.e., inter-cluster arguments).

Our Approach. In order to address the deficiencies mentioned above, we present an approach which trains a generative model to simultaneously utilize the relationships between intra-cluster and inter-cluster arguments. This model generates a key point for a pair of inter-cluster arguments and provided a score indicating whether there exists shared key points between the pair of arguments. During the prediction phase, we utilize the score provided by the generative model, which indicates whether a pair of arguments is inter-cluster arguments, rather than the semantic similarity of the arguments, to partition the arguments into multiple clusters. This score is provided by a fine-tuned generative model, which is aligned with the data distribution within the domain, rather than employing a clustering model that is unrelated to the domain. We then construct an argument graph and formulate the relationships among arguments as a graph-based problem. Specifically, we represent arguments as vertices in the graph, the shared key point between two arguments as edges, and the scores as edge weights, resulting in a weighted argument graph. We introduce an iterative graph partitioning algorithm, which generates several subgraphs, each representing a collection of arguments that share the same or similar key point. From these subgraphs, we select a representative key point for further analysis and utilization.

To summarize, our contributions include

- proposing a pairwise generation approach that simultaneously utilizes the information between intra-cluster arguments and inter-

cluster arguments and providing a score to indicate the presence of the shared key point between a pair of arguments, and

- introducing a weighted graph partition algorithm designed specifically for the KPA task. Our algorithm evaluates the connections between arguments by the scores provided by the generative model, as semantic similarity may not fully align with the objectives of the KPA task.

Outline. We elaborate our approach in Section 2, present experiments in Section 3, discuss related work in Section 4, and conclude in Section 5.

Code. Our code and data are available on GitHub: https://github.com/Alibaba-NLP/Key-Point-Analysis.

2 Approach

On the KPA task, the input consists of a set of arguments $\mathbb{A} = \{(\text{arg}_1, \text{stance}_1), (\text{arg}_2, \text{stance}_2), \cdots\}$ under a given topic T, where arg_i is the argument text and stance_i is the stance for arg_i. The stance ("pro" or "con") indicate the stance of argument towards the topic. The ideal outcome is a concise set of key points $\mathbb{K} = \{\text{kp}_1, \text{kp}_2, \cdots\}$, where each key point in $\mathbb{K}$ corresponds to one or more arguments in $\mathbb{A}$. That is, our objective is to learn a surjective function $f : \mathbb{A}, T \to \mathbb{K}$. Afterwards, the importance of each key point can be expressed by calculating the percentage of arguments corresponding to each key point.

Figure 2 provides an overview of our proposed method, which comprises two principal components. The initial component is dedicated to the generation of key points, as depicted at the bottom of Figure 2. During this phase, we process the given topic T and pairs of arguments $\mathbb{A} \times \mathbb{A}$ to ascertain if each pair shares a key point and, if so, generate the corresponding key point. The subsequent component entails graph partitioning, illustrated at the top of Figure 2. In this stage, we construct a graph of arguments, key points, and their associated scores. We then apply our specially developed graph partitioning algorithm to divide the graph into multiple subgraphs, allowing for the possibility of the same argument being present in more than one subgraph, as a single argument may relate to multiple key points. Finally, we select a representative key point from each subgraph to
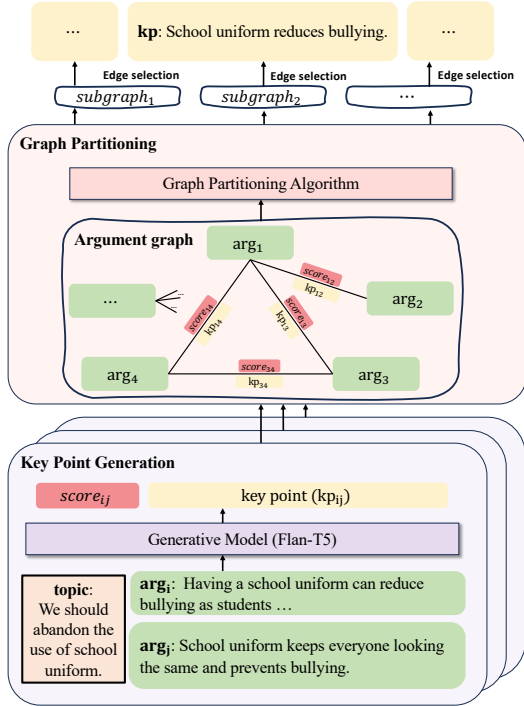
Figure 2: Overview of our approach. Each argument possesses a stance representing its polarity ("pro" or "con"). For the sake of brevity, it has been omitted in the figure.

form the final concise set of key points $\mathbb{K}$. Notably, unlike previous pipeline approaches, we transmit the scores generated by the first stage to the second stage, avoiding issues such as feature propagation interruption and error accumulation that may arise in multi-stage pipelines.

2.1 Key Point Generation

Inspired by contrastive learning (Khosla et al., 2020; Gao et al., 2021), our approach takes a fine-grained perspective by pairing arguments and determining whether a pair of arguments shares the same key point. We posit that it is equally important to model the relationships between inter-cluster arguments that possess different key points and intra-cluster arguments that share a key point. The previous models either mapped each argument to a single key point (Alshomary et al., 2021; Kapadnis et al., 2021) or clustered arguments together and concatenated multiple arguments within a cluster to generate a key point (Li et al., 2023).

Construction of Training Data. We pack a pair of arguments into an input text

$$\texttt{input}_{ij} = \texttt{topic} \mid \texttt{stance}_i \ \texttt{arg}_i \mid \texttt{stance}_j \ \texttt{arg}_j,$$

and prefix the output text with a special word to indicate whether the input pair of arguments are intra-cluster arguments or inter-cluster arguments, which is

$$\texttt{output}_{ij} = \begin{cases} \text{Yes. } \{\texttt{kp}_{ij}\}, & \text{for intra-cluster arguments,} \\ \text{No.}, & \text{for inter-cluster arguments.} \end{cases}$$

We construct training data by pairing arguments, disregarding the order between a pair of arguments. That is, we consider $\texttt{input}_{ij}$ while discarding $\texttt{input}_{ji}$ (where $i < j$). We determine whether a pair of arguments are intra-cluster arguments or inter-cluster arguments by assessing if the corresponding key points of the arguments are identical. To avoid overfitting, we limit the occurrence of each argument as an intra-cluster argument in the training data to no more than five times.

input A	We should adopt atheism. | positive. if we adopt atheism then maybe people will start believing in the scientific community again. | positive. we should adopt atheism because science can explain how we got here without needing a god to explain it.
output A	Yes. Science can adequately explain the Universe
input B	We should ban the use of child actors. | positive. child actors do not get to live a proper young persons life and experience childhood play, education and social interaction, so should be banned | positive. the use of child actors should be banned as children are not capable of making important decisions and some do not have trustworthy parents to make rationale decisions.
output B	No.

Table 1: Training examples. The two arguments in example A share a key point, whereas the two arguments in example B do not.

To facilitate a better understanding of the construction of the training data, we present two examples in Table 1.

Generation Model. Subsequently, we feed $\texttt{input}_{ij}$ to the generative model [1],

$$\mathbf{h}_1, \mathbf{h}_2, \cdots, \mathbf{h}_n = \texttt{Decoder}(\texttt{Encoder}(\texttt{input}_{ij})),$$

where $\mathbf{h}_k$ denotes the representation obtained from the Decoder during the k-th decoding step and n is the decoder output length. During the decoding phase, a linear transformation LM_Head is employed to map the representations outputted by the

---

[1] In our paper, we exemplify our approach by employing a model based on the encoder-decoder architecture. However, our approach can also be applied to decoder-only models.

Decoder to the probability distribution for the tokens in the vocabulary at step k, denoted as

$$S_k = \texttt{LM\_Head}(\mathbf{h}_k).$$

The k-th token is decoded by

$$\texttt{token}_k = \texttt{vocab}[\texttt{argmax}(S_k)],$$

where vocab is the vocabulary.

We extract scores for token "Yes" and "No" from S_1 (i.e., $S_1[\text{Yes}]$ and $S_1[\text{No}]$, respesctively) and obtain the score indicating whether $\texttt{arg}_i$ and $\texttt{arg}_j$ sharing the same key point by

$$score_{ij} = \frac{\exp(S_1[\text{Yes}])}{\exp(S_1[\text{Yes}]) + \exp(S_1[\text{No}])}. \quad (1)$$

The loss during the training process is

$$\mathcal{L} = -\frac{1}{n} \sum_{1 \le k \le n} \log \frac{\exp(S_k[\texttt{output}_{ij}[k]])}{\sum_{t \in \text{vocab}} \exp(S_k[t])},$$

where $\texttt{output}_{ij}[k]$ is the k-th token in $\texttt{output}_{ij}$.

By employing this approach, we have integrated the binary classification task of determining the presence of shared key points and the task of generating key points into a single generative model, thereby mitigating potential information loss that may arise from coupling two different models.

2.2 Graph Partitioning

After the generation of key points, it is necessary to aggregate all intra-cluster arguments and determine the shared key point. We transform this issue into a graph partitioning problem, such that each partition yields a subgraph representing a cluster of arguments that share key points. Previous approaches (Li et al., 2023) relied on semantic similarity-based clustering methods. However, it is important to recognize that two arguments sharing the same key point may not always exhibit high semantic similarity, since an argument can encompass multiple key points. An alternative direct approach involves clustering the key points directly. However, the semantic gap between the generative models and the clustering models may introduce performance drop in this approach. We have mitigated this issue by transferring $score_{ij}$ between the two stages. Specifically, in Equation 1, we calculate the score $score_{ij}$ to quantify the presence of a shared key point between argument $\texttt{kp}_i$ and argument $\texttt{kp}_j$. The score $score_{ij}$ is then used to guide the partitioning process.

Argument Graph Construction. We employ arguments $\mathbb{A}$ as vertices and each edge possesses two attributes, an edge weight

$$score \in \mathbb{S} = \{score_{i,j} | 1 \le i, j \le |\mathbb{A}|, i \ne j\},$$

which indicates the probability of the presence of a shared key point, and the generated shared key point

$$\texttt{kp} \in \hat{\mathbb{K}}_{\text{Gen}} = \{\texttt{kp}_{ij} | 1 \le i, j \le |\mathbb{A}|, i < j\}.$$

We have removed the edges for which the model outputs "No", corresponding to the edges between argument vertices that do not share a key point.

In doing so, we construct a weighted argument graph

$$G = \langle \mathbb{A}, (\mathbb{S}, \hat{\mathbb{K}}_{\text{Gen}}) \rangle.$$

Partitioning by Local Search. We employ local search algorithm to partition argument graph. We use the BAAI-bge-large (Xiao et al., 2023) model to obtain representations of arguments, and then apply k-means clustering (Lloyd, 1982) to cluster the arguments, which serves as the initial partitioning for our graph. We then obtain the initial subgraphs $\mathbb{G}_{\text{subgraph}} = \{g_1, g_2, \cdots, g_s\}$, where s is a hyperparameter. Subsequently, we iteratively move vertices across the current subgraphs by optimizing a cost function, until no improvement of cost is possible or the limit of iteration steps denoted by l is reached. We finally obtain subgraphs $\mathbb{G}'_{\text{subgraph}}$ and form the concise set of key points $\mathbb{K}_{\text{Gen}}$ from each subgraph by choosing the key points with the highest edge weights.

Cost Function. We define the weight $\texttt{wt}(g)$ of a subgraph g as the average weight of all edges in g. For a randomly selected subgraph and within it, a randomly selected vertex $\texttt{arg}$, we calculate the change in the graph weight resulting from the movement of vertex $\texttt{arg}$ from the subgraph g_{out} to another subgraph g by our cost function

$$
\begin{aligned}
\texttt{cost}(g&, g_{\text{out}}, \texttt{arg}) = \\
&\texttt{wt}(g_{\text{out}} \setminus \{\texttt{arg}\}) - \texttt{wt}(g_{\text{out}}) + \\
&\texttt{wt}(g \cup \{\texttt{arg}\}) - \texttt{wt}(g).
\end{aligned}
$$

We then search for a target subgraph g_{in} by

$$g_{\text{in}} = \operatorname*{arg\,max}_{g \in \mathbb{G}_{\text{subgraph}} \setminus \{g_{\text{out}}\}} \texttt{cost}(g, g_{\text{out}}, \texttt{arg}),$$

which means that moving the selected vertex $\texttt{arg}$ from g_{out} to g_{in} would result in the maximum change in graph weight and the selected vertex will only be moved when the weight change is positive.

Soft Partition. Furthermore, since an argument may encompass multiple key points, it is permissible to allow an argument to belong to multiple subgraphs simultaneously, which we refer to as soft partition. We set a threshold h, such that if the removal of the selected argument from its original subgraph would cause the weight of the subgraph to decrease beyond the threshold h, we will retain the argument within the original subgraph. This means that the selected argument can contribute a weight gain greater than h in the original subgraph.

It is worth noting that we have not imposed any restrictions on the connectivity of subgraphs in our algorithm. This is because, given a specified number of subgraphs, we cannot guarantee that all the resulting subgraphs will be connected (as the number of connected components in G may be less than the number of subgraphs). However, it is important to note that our optimization objective (i.e., graph weight) tends to favor connected subgraphs or subgraphs with higher connectivity.

3 Experiments

3.1 Datasets

We conducted experiments on ArgKP (Friedman et al., 2021) and QAM (Guo et al., 2023). ArgKP is a KPA dataset annotated by crowd sourcing, with training/validation/test sets consisting of 24/4/3 topics, 5583/932/724 arguments, and corresponding 207/36/33 key points. Each argument and key point possess a stance attribute, either "pro" (i.e., positive) or "con" (i.e., negative), indicating whether they support the current topic. The QAM dataset is originally designed for extracting quadruples $\langle$key point, stance, argument, evidence type$\rangle$ from documents with a given topic. QAM is annotated through crowdsourcing, and we have transformed the QAM dataset into the format required for the KPA task. The training/validation/test sets consists 96/52/53 topics, 6635/808/948 arguments, and corresponding 2366/314/337 key points.

3.2 Evaluation Metrics

We employed Rouge (Lin, 2004) and soft-Precision/Recall/F1 as metrics. Following Li et al., 2023, we obtained Rouge scores by comparing the concatenation of generated key points and the concatenation of reference key points. We reported the scores of Rouge-1 and Rouge-2. For soft-Precision/Recall/F1 (denoted by sP, sR and sF1, respectively), we employed BLEURT (Sellam et al.,

2020) to measure the similarity between individual generated key point and the reference key point. The sP, sR and sF1 to compare the similarity between the sets of key points were given by

$$\mathbf{sP} = \frac{1}{|\mathbb{K}_{\text{Gen}}|} \sum_{\text{kp}_{\text{Gen}} \in \mathbb{K}_{\text{Gen}}} \max_{\text{kp} \in \mathbb{K}} \texttt{BLEURT}(\text{kp}, \text{kp}_{\text{Gen}}),$$

$$\mathbf{sR} = \frac{1}{|\mathbb{K}|} \sum_{\text{kp} \in \mathbb{K}} \max_{\text{kp}_{\text{Gen}} \in \mathbb{K}_{\text{Gen}}} \texttt{BLEURT}(\text{kp}, \text{kp}_{\text{Gen}}) \text{ and}$$

$$\mathbf{sF1} = \frac{2 \times \mathbf{sP} \times \mathbf{sR}}{\mathbf{sP} + \mathbf{sR}}.$$

3.3 Implement Details

We conducted experiments on NVIDIA A100 (80GB) GPUs. Our implementations were based on PyTorch 2.0.1 and HuggingFace Transformers 4.30.2. We set *learning rate* $= 1e - 5$ selected from $\{1e - 5, 3e - 5, 6e - 5\}$, *batch size* $= 64$ selected from $\{32, 64, 128, 256\}$, *random seed* $= 42$, *maximum sequence length* $= 512$, *epoch* $= 5$. We set $h = 0.008$ for ArgKP, $h = 0.006$ for QAM and $l = 200$ for ArgKP, $l = 80$ for QAM. In order to ensure a fair comparison with previous work (Li et al., 2023), we set the number of selected key points, which is the number of subgraphs s, to be equal to the number of reference key point set.

3.4 Previous State-of-the-Art Approaches

We compared with three state-of-the-art models: graph based summarization (GBS, Alshomary et al., 2021), Enigma (Kapadnis et al., 2021) and supervised key point modelling (SKPM, Li et al., 2023). The GBS model was a representative of the extractive key point generation models. It employed arguments to establish a graph, where each argument vertex was examined to determine their suitability as a key point. The PageRank (Page et al., 1999) algorithm was then utilized to choose the key points. Enigma generated a key point for each argument based on PEGASUS (Zhang et al., 2020) and selected the highest-scoring key point using the precision of Rouge-1 compared with reference key point set. SKPM was a two-stage model that first clusterd arguments and then generated key points for arguments within each cluster.

3.5 Comparison with Previous State-of-the-Art Approaches

In comparison with the existing methods as shown in Table 2, our approach demonstrated advantages over the previous SOTA approaches. Specifically, on the ArgKP dataset, we observed improvements

Model	ArgKP					QAM				
	Rouge-1	Rouge-2	sP	sR	sF1	Rouge-1	Rouge-2	sP	sR	sF1
Previous SOTA Approaches										
GBS	19.60	3.40	53.00	52.00	52.00	-	-	-	-	-
Enigma$_{PEGASUS}$	20.00	4.80	58.00	57.00	57.00	**32.61**	7.17	46.96	<u>44.43</u>	45.51
SKPM$_{Flan-T5-base}$	30.30	8.90	59.00	58.00	59.00	-	-	-	-	-
SKPM$_{Flan-T5-large}$	31.40	9.10	57.00	**62.00**	<u>60.00</u>	-	-	-	-	-
Ours										
w/ Flan-T5-base	<u>37.38</u>	<u>11.39</u>	<u>61.29</u>	58.08	59.59	30.31	<u>7.44</u>	48.62	43.81	<u>45.96</u>
w/ Flan-T5-large	**50.91**	**21.23**	**62.57**	<u>60.26</u>	**61.34**	<u>30.44</u>	**8.02**	**49.38**	**45.27**	**47.11**

Table 2: Comparison with previous state-of-the-art approaches. The best results are in bold and the second best results are underlined. The dash signifies that we did not obtain the results due to issues such as code availability. The results of previous state-of-the-art approaches on ArgKP are taken from Li et al., 2023.

of 19.51–31.31 points in Rouge-1 and 1.34–9.34 points in sF1. On the QAM dataset, our approach yielded an improvement of 1.60 points in sF1. The results demonstrated the effectiveness of our method, which were analyzed in detail in the subsequent sections.

3.6 Effectiveness of Key Point Generation

We conducted several experiments to evaluate the effectiveness of our generative model in identifying and generating the shared key point between a pair of arguments. These experiments were structured around two research sub-questions. **SQ1** (Shared Key Point Detection) focused on the ability of our model to determine the presence of a shared key point, which was essentially a binary classification task. **SQ2** (Key Point Generation) assessed the model's capability to generate high-quality key points that effectively covered the reference key point set. In tackling these research sub-questions, we prioritized the recall metric because our subsequent step involves selecting a concise subset from an extensive set of key points, as explained in Section 3.7).

3.6.1 SQ1: Shared Key Point Detection

We applied a representative model RoBERTa-large (Liu et al., 2019) to classify whether two arguments shared a key point and compared its performance with that of our generative model. In our generative model, if a key point was generated for a pair of arguments, the output was classified as positive (i.e., the output began with "Yes"). Conversely, if no key point was generated, the output was classified as negative (i.e., the output began with "No"). For the RoBERTa-large model, we utilized the CLS vector, followed by a linear layer

that produced two scores, one for positive and one for negative, to serve as the classifier.
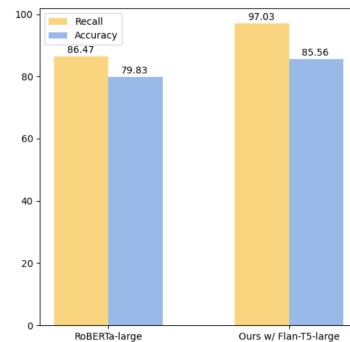

Figure 3: Shared key point detection on ArgKP.

We evaluated the recall and accuracy of two models in a binary classification task aimed at identifying whether a pair of arguments shared a key point on ArgKP. As indicated in Figure 3, our approach outperformed by 10.56 points in recall, the metric of greater significance in our scenario. Generating key points may also help in determining whether there are shared key points among arguments. With pairwise generation, despite the Flan-T5-large being an encoder-decoder model primarily intended for generative tasks, it still delivered satisfactory performance in the classification task.

3.6.2 SQ2: Key Point Generation

In this section, we designed an experiment to verify that the relationship between a pair of intra-cluster arguments with shared key points and inter-cluster arguments without shared key points is beneficial for key point generation. We removed the pairs of arguments that did not share key points from our training data as ablation experiment, which is the result of the "Ours w/o inter-cluster argument pairs" row in Figure 4. To verify the effectiveness of pair-

wise generation, we trained a Flan-T5-large model with the concatenation of the topic, stance and a single argument (i.e., input = topic | stance | arg) as the input, and an output of concatenated key points implied by the single argument. We denoted this approach as "One2KP". Here, we also focused on the recall metric (i.e., sR), which reflected the coverage of the reference key point set during the key point generation stage.
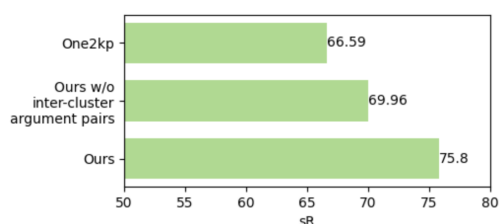

Figure 4: Comparison of different models on key point generation on ArgKP. All the models were based on Flan-T5-large.

From Figure 4, it can be seen that our method had an advantage over the One2KP method, leading by 9.21 points on the sR metric. After separating inter-cluster argument pairs from the training set, a decrease of 5.84 point on the sR metric was observed. Experiments in Figure 4 validated that generating key points from pairwise arguments was effective. It was not only necessary to consider the relationships among intra-cluster arguments, but also inter-cluster arguments.

Additionally, addressing two related tasks simultaneously within the same model did not introduce a drop on the model performance. In our task, as seen in Figure 3 and Figure 4, the carefully designed tasks and the use of pairwise generations method brought about a gain in model performance. This allowed our model to not only exhibit superior performance in key point generation but also achieve satisfactory discriminative ability in intra-cluster arguments and inter-cluster arguments classification.

3.7 Effectiveness of Graph Partitioning

In this section, we aimed to validate the effectiveness of our graph partitioning method. We compared our graph partitioning method with existing clustering methods and graph partitioning methods in Section 3.7.1. Additionally, we investigated the influence of the number of iterations of our graph partitioning method on the performance in Section 3.7.2.

3.7.1 Comparison with Clustering and Graph Partitioning Methods

We substituted our graph partitioning algorithm with either clustering methods or graph partitioning methods. Our key point generation model, exemplified by the Flan-T5-large model, produced key points for these methods and provided scores to determine if two arguments shared a key point for graph partitioning methods. The clustering algorithms we compared were of two types, one clustered arguments to obtain argument subgraphs and then selected key points with highest edge weights, and the other directly clustered key points and then selected the key point at the center of the cluster. We utilized well-established clustering algorithms, namely k-means (Lloyd, 1982), for our analysis. Two models were utilized to derive representations for arguments: 1) the renowned RoBERTa-large (Liu et al., 2019) and 2) BAAI-bge-large (Xiao et al., 2023), which is the state-of-the-art model in sentence representation. Furthermore, we employed three well-known graph partitioning methods, including

- Greedy Modularity Maximization (Clauset et al., 2004), which is a graph partitioning algorithm that utilizes a greedy strategy to maximize modularity and identify community structures,

- Louvain Community Detection (Blondel et al., 2008), which is a heuristic method based on modularity optimization by iteratively merging similar vertices to uncover community structures, and

- Asynchronous Label Propagation (Raghavan et al., 2007), which relies on label propagation iterations to divide the graph into communities by updating vertex labels.

From Table 3, our model exhibited superior performance compared to the three sets of baseline models on Rouge-2, sR and sF1 metrics. Specifically, on the ArgKP dataset, our model exhibited a lead of 0.60–6.27 points in sF1. On the QAM dataset, our model demonstrated a lead of 0.58–11.88 points in sF1.

Our approach amalgamated the strengths of both by identifying shared key points through our generative model, taking into account semantic information, while simultaneously utilizing graph structure information via the graph partitioning method, resulting in improved outcomes.

Model	ArgKP					QAM				
	Rouge-1	Rouge-2	sP	sR	sF1	Rouge-1	Rouge-2	sP	sR	sF1
Clustering Methods (on arguments)										
k-means$_{RoBERTa-large}$	50.70	21.01	59.38	59.49	59.34	29.30	7.26	48.30	44.67	46.28
k-means$_{BAAI-bge-large}$	49.01	18.64	60.94	60.08	60.47	30.13	7.85	47.75	44.99	46.21
Clustering Methods (on key points)										
k-means$_{RoBERTa-large}$	52.71	20.16	61.30	60.21	60.74	28.28	6.77	48.35	43.48	45.67
k-means$_{BAAI-bge-large}$	51.75	19.57	61.42	58.80	59.99	29.40	7.50	48.79	44.70	46.53
Graph Partitioning Methods										
Greedy Modularity Maximization	40.85	14.75	56.15	54.04	55.07	18.59	4.79	39.34	33.85	36.19
Louvain Community Detection	43.01	18.03	56.81	56.65	56.69	17.96	4.68	38.28	32.97	35.23
Asynchronous Label Propagation	17.32	5.54	63.76	53.31	58.01	14.71	4.41	54.40	40.62	46.04
Ours	50.91	21.23	62.57	60.26	61.34	30.44	8.02	49.38	45.27	47.11

Table 3: Comparison with clustering and graph partitioning methods with Flan-T5-large. The best results are in bold and the second best results are underlined.

3.7.2 Analysis of Iterations Steps (l)

#steps	Rouge-1	Rouge-2	sP	sR	sF1
0	49.01	18.64	60.94	60.08	60.47
50	51.46	20.55	61.38	60.63	60.96
100	47.53	19.77	60.92	59.25	60.03
150	52.13	22.39	61.22	60.52	60.85
200	50.91	21.23	62.57	60.26	61.34
250	49.23	19.45	62.32	59.93	61.09

Table 4: Analysis of iterations steps (l).

We conducted an analysis on the impact of the number of iterations steps in the graph partition algorithm. When the number of steps equaled 0, it represented a ablation experiment, indicating the exclusion of any execution of the graph partition algorithm. As shown in Table 4, it could be seen that, excluding the subtle fluctuations in the data, the overall trend of sF1 was first an increase and then a stabilization. Additionally, it could be seen that as the algorithm iterations progress, sR did not change much overall, and our graph partitioning algorithm improved the overall performance by improving sP.

4 Related Work

4.1 Key Point Analysis

Unlike multi-document summarization (Mani and Bloedorn, 1997; Xiao et al., 2022), KPA (Bar-Haim et al., 2020a) requires the production of a concise collection of key points rather than a cohesive paragraph. Furthermore, KPA entails a quantitative analysis of these key points, involving quantifying their prevalence within the arguments.

Existing methods for key points generation in KPA rely on extractive summarization and abstractive summarization techniques. Extractive summa-

rization methods (Bar-Haim et al., 2020b, 2021; Alshomary et al., 2021) involve selecting short and representative arguments as key points. However, extractive summarization approaches are limited by the quality of the arguments and do not ensure the presence of an optimal key point within the arguments. On the other hand, abstractive summarization methods utilize generative models to produce key points for the arguments. Kapadnis et al., 2021 generates key points for each argument individually and then select a collection of key points based on the Rouge (Lin, 2004) scores. Li et al., 2023 first employs clustering methods to group semantically similar arguments into the same cluster, and then concatenates the arguments within the cluster to generate key points.

In contrast, our approach simultaneously takes into account the relationships between arguments with shared key points and those without shared key points. Additionally, by replacing semantic similarity scores with scores for the presence of shared key points between arguments, we address the shortcomings of existing models.

4.2 Graph Partitioning

Unlike clustering methods which depends on the similarity or distance of clustering objects, graph partitioning aims to optimize a certain metric of the resulting subgraphs (Buluç et al., 2013), which is the probability that the arguments within a common subgraph share a common key point in the KPA scenario. The algorithms (Kernighan and Lin, 1970; Krauthgamer et al., 2009; Margo and Seltzer, 2015) based on optimizing the minimum vertex/edge cuts has been extensively utilized. Additionally, algorithms (Clauset et al., 2004; Blondel et al., 2008; Dugué and Perez, 2015; Traag et al., 2018) have

approached graph partitioning by optimizing the modularity of subgraphs. Another avenue (Ragha-van et al., 2007; Cordasco and Gargano, 2010) involves achieving graph partitioning through vertex labels propagation.

In the KPA task, our objective is to maximize the probability that arguments within a subgraph share the same key point while allowing arguments that contain more than one key point to exist in multiple subgraphs simultaneously. Given this requirement, the aforementioned graph partitioning algorithms are not suitable for the KPA task.

5 Conclusion

This paper presents a method that merges pairwise generation with graph partitioning to efficiently produce key points within arguments. Our integrated approach models both the relationships between arguments sharing key points and those that do not, aiding in key point generation. We employ an iterative algorithm to partition the argument graph and extract representative key points from subgraphs, yielding a concise set of key points. In the future, we will continue to explore better ways to utilize generative models to model the relationships among arguments. For instance, we will investigate improved methods of outputting scores indicating whether arguments share key points and develop graph partition algorithms that better align with generative models. Our approach may provide inspiration for tasks including clustering tasks based on generative models (Zhang et al., 2023; Viswanathan et al., 2023), set generation tasks (Madaan et al., 2022), and so forth.

Limitations

The input consists of pairwise arguments, which increases the amount of training data and the computational cost of training. Currently, training time of Flan-T5-large (780M) on ArgKP is approximately 12 hours, and on QAM it is about 6 hours. Our model does not fully utilize the graph structure during the graph partitioning process, and we plan to incorporate more graph structural information into the model in the future. The datasets currently used for experiments are related to survey review topics, and there is a lack of datasets from other fields (such as product reviews in the e-commerce scenario) in the research community. Future research could explore the effectiveness on datasets from different domains.

References

Milad Alshomary, Timon Gurcke, Shahbaz Syed, Philipp Heinisch, Maximilian Spliethöver, Philipp Cimiano, Martin Potthast, and Henning Wachsmuth. 2021. Key point analysis via contrastive learning and extractive argument summarization. In *Proceedings of the 8th Workshop on Argument Mining*, pages 184–189, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Roy Bar-Haim, Lilach Eden, Roni Friedman, Yoav Kantor, Dan Lahav, and Noam Slonim. 2020a. From arguments to key points: Towards automatic argument summarization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4029–4039, Online. Association for Computational Linguistics.

Roy Bar-Haim, Lilach Eden, Yoav Kantor, Roni Friedman, and Noam Slonim. 2021. Every bite is an experience: Key Point Analysis of business reviews. 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 3376–3386, Online. Association for Computational Linguistics.

Roy Bar-Haim, Yoav Kantor, Lilach Eden, Roni Friedman, Dan Lahav, and Noam Slonim. 2020b. Quantitative argument summarization and beyond: Cross-domain key point analysis. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 39–49, Online. Association for Computational Linguistics.

Vincent D. Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. 2008. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2008:P10008.

Aydın Buluç, Henning Meyerhenke, Ilya Safro, Peter Sanders, and Christian Schulz. 2013. Recent advances in graph partitioning. In *Algorithm Engineering*.

Arie Cattan, Lilach Eden, Yoav Kantor, and Roy Bar-Haim. 2023. From key points to key point hierarchy: Structured and expressive opinion summarization. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 912–928, Toronto, Canada. Association for Computational Linguistics.

Aaron Clauset, Mark E. J. Newman, and Cristopher Moore. 2004. Finding community structure in very large networks. *Physical review. E, Statistical, nonlinear, and soft matter physics*, 70 6 Pt 2:066111.

Gennaro Cordasco and Luisa Gargano. 2010. Community detection via semi-synchronous label propagation algorithms. In *2010 IEEE International Workshop on: Business Applications of Social Network Analysis (BASNA)*, pages 1–8.

Nicolas Dugué and Anthony Perez. 2015. Directed louvain : maximizing modularity in directed networks.

Roni Friedman, Lena Dankin, Yufang Hou, Ranit Aharonov, Yoav Katz, and Noam Slonim. 2021. Overview of the 2021 key point analysis shared task. In *Proceedings of the 8th Workshop on Argument Mining*, pages 154–164, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Tianyu Gao, Xingcheng Yao, and Danqi Chen. 2021. SimCSE: Simple contrastive learning of sentence embeddings. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 6894–6910, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Jia Guo, Liying Cheng, Wenxuan Zhang, Stanley Kok, Xin Li, and Lidong Bing. 2023. AQE: Argument quadruplet extraction via a quad-tagging augmented generative approach. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 932–946, Toronto, Canada. Association for Computational Linguistics.

Manav Kapadnis, Sohan Patnaik, Siba Panigrahi, Varun Madhavan, and Abhilash Nandy. 2021. Team enigma at ArgMining-EMNLP 2021: Leveraging pre-trained language models for key point matching. In *Proceedings of the 8th Workshop on Argument Mining*, pages 200–205, Punta Cana, Dominican Republic. Association for Computational Linguistics.

B. W. Kernighan and S. Lin. 1970. An efficient heuristic procedure for partitioning graphs. *The Bell System Technical Journal*, 49(2):291–307.

Prannay Khosla, Piotr Teterwak, Chen Wang, Aaron Sarna, Yonglong Tian, Phillip Isola, Aaron Maschinot, Ce Liu, and Dilip Krishnan. 2020. Supervised contrastive learning. In *Advances in Neural Information Processing Systems*, volume 33, pages 18661–18673. Curran Associates, Inc.

Robert Krauthgamer, Joseph (Seffi) Naor, and Roy Schwartz. 2009. Partitioning graphs into balanced components. In *Proceedings of the Twentieth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '09, page 942–949, USA. Society for Industrial and Applied Mathematics.

Hao Li, Viktor Schlegel, Riza Batista-Navarro, and Goran Nenadic. 2023. Do you hear the people sing? key point analysis via iterative clustering and abstractive summarisation. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 14064–14080, Toronto, Canada. Association for Computational Linguistics.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *ArXiv*, abs/1907.11692.

Stuart P. Lloyd. 1982. Least squares quantization in pcm. *IEEE Trans. Inf. Theory*, 28:129–136.

Aman Madaan, Dheeraj Rajagopal, Niket Tandon, Yiming Yang, and Antoine Bosselut. 2022. Conditional set generation using seq2seq models. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 4874–4896, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Inderjeet Mani and Eric Bloedorn. 1997. Multi-document summarization by graph search and matching. In *AAAI/IAAI*.

Daniel Margo and Margo Seltzer. 2015. A scalable distributed graph partitioner. *Proc. VLDB Endow.*, 8(12):1478–1489.

Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. 1999. The pagerank citation ranking : Bringing order to the web. In *The Web Conference*.

Usha Nandini Raghavan, Réka Albert, and Soundar Kumara. 2007. Near linear time algorithm to detect community structures in large-scale networks. *Phys. Rev. E*, 76:036106.

Thibault Sellam, Dipanjan Das, and Ankur P Parikh. 2020. Bleurt: Learning robust metrics for text generation. In *Proceedings of ACL*.

Vincent Antonio Traag, Ludo Waltman, and Nees Jan van Eck. 2018. From louvain to leiden: guaranteeing well-connected communities. *Scientific Reports*, 9.

Vijay Viswanathan, Kiril Gashteovski, Carolin (Haas) Lawrence, Tongshuang Sherry Wu, and Graham Neubig. 2023. Large language models enable few-shot clustering. *ArXiv*, abs/2307.00524.

Shitao Xiao, Zheng Liu, Peitian Zhang, and Niklas Muennighoff. 2023. C-pack: Packaged resources to advance general chinese embedding. *ArXiv*, abs/2309.07597.

Wen Xiao, Iz Beltagy, Giuseppe Carenini, and Arman Cohan. 2022. PRIMERA: Pyramid-based masked sentence pre-training for multi-document summarization. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5245–5263, Dublin, Ireland. Association for Computational Linguistics.

Jingqing Zhang, Yao Zhao, Mohammad Saleh, and Peter J. Liu. 2020. Pegasus: Pre-training with extracted gap-sentences for abstractive summarization. In *Proceedings of the 37th International Conference on Machine Learning*, ICML'20. JMLR.org.

Yuwei Zhang, Zihan Wang, and Jingbo Shang. 2023. Clusterllm: Large language models as a guide for text clustering. *ArXiv*, abs/2305.14871.